Clube Delphi +PHP

DevMedia group

Ano 9 · Edição 103 · R$11,90

DataSnap & SOAP

Integração fácil, rápida e prática de suas aplicações na Web

Ask the Expert
Compartilhando sem permissão no Windows

QuickUpdate
Aprenda a programar brincando e em 3D

DataSnap e SOAP
Crie aplicações integradas com duas das principais tecnologias existentes no mercado

Assistentes passo a passo na Web
Crie Wizards para as mais diversas situações em aplicações Web

EasyDelphi
Crie seu próprio processador de textos com o WPTools

PHP
Crie miniaturas de fotografias em suas aplicações

ISSN 1517990-7

Na Web :

Confira no portal ClubeDelphi PLUS **4 vídeo-aulas:**

- Mini-curso: Trabalhando e criando eventos no Delphi – Partes I e II
- Mini-curso: Auto-completar datas digitadas em tela – Partes I e II

Hospede seus sites com os profissionais da Hospedix.

3 DOMÍNIOS PRINCIPAIS

200 DOMÍNIOS HOSPEDADOS

100GB DE TRANSFERÊNCIA MENSAL

30GB PARA E-MAILS

2GB DE ESPAÇO EM DISCO

Clientes Hospedix ainda podem optar por uma de nossas promoções:

1 registro de domínio (.com .net .org ou .info) grátis e isento de taxas anuais.

ou

1 registro de domínio (.com.br) por apenas R$15,00 anuais enquanto for cliente.

Tudo isso por apenas **R$ 25,90** por mês

Assine agora mesmo e descubra por que Hospedix é a hospedagem de sites preferida entre os profissionais.

HOSPEDIX
HOSPEDAGEM PROFISSIONAL
www.hospedix.com.br

30 DIAS GRÁTIS

Ao assinar, digite o código de desconto exclusivo para leitores desta revista e ganhe 30 dias inteiramente grátis.

RVSTMEDIA

As promoções anunciadas não são cumulativas com outras promoções e são válidas apenas para clientes Hospedix ativos.

Sumário

Olá, eu sou o **DevMan**! Desta página em diante, eu estarei lhe ajudando a compreender com ainda mais facilidade o conteúdo desta edição. Será um prazer contar com sua companhia! Confira abaixo o que teremos nesta revista:

Delphi — Tutorial

14 – Sistemas DataSnap com SOAP
Integre seus sistemas utilizando duas das principais tecnologias
[*Maikel Scheid e Edinei Daniel Steffen*]

Delphi — Web, Novidades

24 – ASP.NET 2.0
Crie assistentes inteligentes na internet com o controle Wizard
[*Adriano Santos*]

Boas Práticas — Boas Práticas, Projeto/Análise

28 – Padrões de Projeto
Entenda o que são Design Patterns e como aplicá-los a seu projeto
[*Paulo Quicoli*]

Boas Práticas — Boas Práticas, Projeto/Análise

32 – Desenvolvimento em equipe
Controle e administre o trabalho em equipe com o Subversion
[*Paulo Quicoli*]

Easy Delphi — Easy Delphi

40 – Criando um editor de textos com o WPTools
Conheça os componentes WPTools e desenvolva seu próprio processador de textos
[*Paulo Quicoli*]

PHP — PHP

46 – Thumbnails em sites PHP
Veja como criar miniaturas de fotos em PHP
[*Adriano Santos*]

Você percebeu os ícones ao lado de cada matéria? Eles indicam o que você vai encontrar no artigo – dessa forma, você também pode ter uma idéia geral do que vai encontrar nesta edição como um todo! Os editores trabalham sempre no sentido de fechar a revista seguindo esta definição, para oferecer a você o melhor conteúdo didático!

Confira abaixo a lista com a definição dos tipos de artigo encontrados nesta edição:

[Mão na Massa] Artigos que focam na resolução de problemas - e não na tecnologia. A tecnologia empregada é apenas consequência. O artigo parte do pressuposto que o leitor já conhece a tecnologia, mas tem dificuldade de implementá-la em aplicação cotidiana. Por exemplo, um artigo que ao invés de "debulhar" todos os métodos e propriedades de um "recordset", mostre como usar o recordset de forma inteligente para criar um carrinho de compras.

[PHP] Artigo com foco no PHP puro.

[Expert] Artigo com foco no leitor avançado.

[Boas Práticas] Um dos objetivos da revista é levar para o leitor não somente as melhores técnicas mas também as melhores práticas. Esse tipo de artigo foca em técnicas que poderão aumentar a qualidade do desenvolvimento de software.

[Web] Artigos sobre/ ou que envolvam técnicas de desenvolvimento para Web.

[Novidades] Artigos sobre tecnologias de ponta, que ainda não fazem parte o dia-a-dia do desenvolvedor mas que prometem ser a próxima febre.

[Boa Idéia] A esse tipo de artigo é quase um artigo "Mão na Massa" - a diferença é que o exemplo do artigo é tão criativo que é considerado uma "Boa Idéia", dentro do contexto do desenvolvimento de software.

[Easy Delphi] Com foco no desenvolvedor iniciante.

Clube Delphi +PHP

Ano 9 - 103ª Edição 2009 - ISSN 1517990-7 Impresso no Brasil

Corpo Editorial

Editor Geral
Adriano Santos
adrianosantos@devmedia.com.br

Editor Técnico
Paulo Quicoli
pauloquicoli@devmedia.com.br

Comissão Editorial
Fabrício Desbessel, Paulo Quicoli, Adriano Santos e Guinther Pauli.

Editor de Arte
Vinicius O. Andrade
viniciusoandrade@gmail.com

Diagramação
Romulo Araujo
romulo@devmedia.com.br

Revisão
Gregory Monteiro - gregory@clubedelphi.net
Thiago Vincenzo - thiago.v.ciancio@devmedia.com.br

Distribuição
Fernando Chinaglia Dist. S/A
Rua Teodoro da Silva, 907
Grajaú - RJ - 206563-900

Atendimento ao Leitor

A DevMedia conta com um departamento exclusivo para o atendimento ao leitor. Se você tiver algum problema no recebimento do seu exemplar ou precisar de algum esclarecimento sobre assinaturas, exemplares anteriores, endereço de bancas de jornal, entre outros, entre em contato com:

Carmelita Mulin – Atendimento ao Leitor
www.devmedia.com.br/central/default.asp
(21) 3382-5025

Kaline Dolabella – Gerente de Marketing e Atendimento
kalined@terra.com.br
(21) 3382-5025

Publicidade

Para informações sobre veiculação de anúncio na revista ou no site e para fechar parcerias ou ações específicas de marketing com a DevMedia, entre em contato com:

Kaline Dolabella
publicidade@devmedia.com.br

Fale com o Editor!

É muito importante para a equipe saber o que você está achando da revista: que tipo de artigo você gostaria de ler, que artigo você mais gostou e qual artigo você menos gostou. Fique a vontade para entrar em contato com os editores e dar a sua sugestão!
Se você estiver interessado em publicar um artigo na revista ou no site ClubeDelphi, entre em contato com os editores, informando o título e mini-resumo do tema que você gostaria de publicar:

Guinther Paulli - Editor da Revista
guinther@devmedia.com.br

EDITORIAL

Nunca se falou tanto em integração de aplicações como em 2008 em minha opinião. Permitir que um mesmo banco de dados e suas regras de negócio fossem acessadas e processadas em diferentes plataformas, tais como: Desktop, Internet e dispositivos móveis, passou a ser a meta de muitos gerentes de TI ao redor do mundo. Tudo isso se deve ao fato de que a Internet não é mais aquela "menininha tímida" de alguns anos atrás. Hoje, a Internet está mais presente na vida de todos como nunca esteve e certamente isso é apenas o começo de uma grande fase em nossas vidas.

Imagine desenvolver uma aplicação que pode ser utilizada em diversos dispositivos, incluindo telefonia móvel, PocketPC's, Palms e BlackBerries sem precisar de grandes modificações no código-fonte? Realmente isso parece coisa de filme, mas vos digo que é possível. Nessa edição de ClubeDelphi veremos um excelente artigo de Maikel Scheid que ensina como criar uma aplicação bastante robusta e inteligente o bastante para ser utilizada em qualquer plataforma usando duas grandes tecnologias: DataSnap e SOAP. O artigo mostra a criação de WebServices que torna externas as principais funcionalidades do sistema, fazendo com que qualquer pessoa possa desenvolver soluções para leitura, inclusão, alteração e até processamento de informações no banco de dados.

Na seqüência veremos dois artigos do Paulo que visam as Boas Práticas do desenvolvimento. No primeiro artigo, o Paulo fala sobre o desenvolvimento em equipe utilizando o Subversion, que controla o versionamento do código-fonte e as atribuições de cada um dentro do projeto. Em seguida, veja como trabalhar com os principais padrões de projeto para tornar sua aplicação e as regras do desenvolvimento mais fáceis e ágeis de lidar.

Na seção EasyDelphi, o Paulo demonstra todo o poder dos componentes WPTools. Com eles é possível criar nosso próprio editor de textos com as mais variadas opções, como Negrito, Itálico, Sublinhado, alinhamentos e tudo que podemos imaginar de um editor.

Sabe aquelas miniaturas de imagens que aparecem em diversos sites, como Orkut, por exemplo? Pois bem, aquilo chama-se de Thumbnails. É perfeitamente possível criar esse recurso utilizando PHP. No meu artigo, explico como desenvolver esse recurso fazendo a criação de um pequeno álbum de fotografias com o recurso de Thumbnails.

Em 2009, muito sucesso para todos nós. Até a próxima.

Adriano Santos

adrianosantos@devmedia.com.br

Editor Chefe da revista Clube Delphi
Conheça nosso BLOG em clubedelphiphp.blogspot.com

ClubeDelphi PLUS

DevMedia group

A revista ClubeDelphi é parte integrante da assinatura ClubeDelphi PLUS.
Para mais informações sobre o pacote PLUS, acesse:
http://www.devmedia.com.br/clubedelphi/portal.asp

Hospede com quem fala sua língua!

Local host

Hospedagem Profissional
Revenda
Parcerias

Ativação imediata!

Faça sua assinatura, tenha **30 dias grátis**

e seu primeiro boleto para **60 dias após assinatura** em qualquer plano

UNIX I

a partir **R$ 28,00**

- 1 GB Espaço em disco
- 60 GB Transferência
- 30 Contas de e-mail
- 10 Domínios
- Painel de Controle em Português
- Atendimento exclusivo

Contratar

UNIX II

a partir **R$ 58,00**

- 3 GB Espaço em disco
- 120 GB Transferência
- 60 Contas de e-mail
- 15 Domínios
- Painel de Controle em Português
- Atendimento exclusivo

Contratar

UNIX III

a partir **R$ 98,00**

- 6 GB Espaço em disco
- 300 GB Transferência
- 90 Contas de e-mail
- 20 Domínios
- Painel de Controle em Português
- Atendimento exclusivo

Contratar

R$ 5,90 — **Mini html**
Tráfego 1 GB
Espaço 100 MB
3 C.Ppostal, Espaço 3G
Domínio 3

R$ 15,90 — **html**
Tráfego 25 GB
Espaço 500 MB
20 C.Ppostal, Espaço 20G
Domínio Ilimitado

R$ 15,90 — **ASP**
Tráfego 25 GB
Espaço 500 MB
20 C.Ppostal, Espaço 20G
Domínio Ilimitado
MySql 50m

R$ 20,90 — **PHP**
Tráfego 25 GB
Espaço 500 MB
20 C.Ppostal, Espaço 20G
Domínio Ilimitado
MySql 50m

Formato exclusivo de trabalho para empresas de desenvolvimento de sites, websites, portais, serviços e negócios que necessitam de grandes tráfegos, monitoramento e backup.

www.lhost.com.br

assinar@lhost.com.br

ClubeDelphi PLUS

Informativo ClubeDelphi

Portal ClubeDelphi

www.clubedelphi.net/portal

+ 600 vídeo aulas e 7 cursos online

Brinde na web desta edição 4 vídeos com temas completos!

4 Vídeos

• **Mini-curso: Trabalhando com eventos na prática - Parte 1 e 2**
Domine a utilização de eventos na prática com o mini-curso do colunista Renato Matos. Você aprenderá a utilizar os já existentes na VCL, bem com a criação de seus próprios eventos na aplicação.

http://www.devmedia.com.br/articles/viewcomp.asp?comp=10512
http://www.devmedia.com.br/articles/viewcomp.asp?comp=10625

• **Como auto-completar uma data digitada por um usuário - Parte 1 e 2**
Veja como criar um controle que permita a digitação parcial de uma data e a implementação de um interessante recurso para auto-completar as datas digitadas.

http://www.devmedia.com.br/articles/viewcomp.asp?comp=9248
http://www.devmedia.com.br/articles/viewcomp.asp?comp=9329

Para acessar os vídeos utilize os seguintes dados
Login: ***DVM.CD*** *Senha:* ***xbd89dp***

Gostou das vídeo aulas? O portal *www.devmedia.com.br* possui mais de **2 mil vídeo aulas** e **dezenas de cursos online** sobre desenvolvimento de software! Agora você pode comprar as vídeo aulas que preferir e fazer sua própria combinação de vídeos! Saiba mais em *www.devmedia.com.br/creditos*

Edições Anteriores da ClubeDelphi

Você pode comprar todas as edições anteriores da ClubeDelphi através do site da DevMedia!

Para isso basta acessar *https://seguro.devmedia.com.br/edicoes_anteriores.asp*

Dê seu feedback sobre esta edição!

A ClubeDelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre esta edição, artigo por artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Para votar, você vai precisar do código e senha de banca desta edição, que é:
Login: **DVM.CD** Senha: **xbd89dp**

Conhecimento faz diferença!

Mais de 60 mil downloads na primeira edição!

Faça já sua assinatura digital! | *www.devmedia.com.br/es*

Faça um *up grade* em sua carreira.

Em um mercado cada vez mais focado em qualidade ter conhecimentos aprofundados sobre requisitos, metodologia, análises, testes, entre outros, pode ser a diferença entre conquistar ou não uma boa posição profissional. Sabendo disso a DevMedia lança para os desenvolvedores brasileiros sua primeira revista digital totalmente especializada em Engenharia de Software. Todos os meses você irá encontrar artigos sobre Metodologias Ageis; Metodologias tradicionais (*document driven*); ALM (*application lifecycle management*); SOA (aplicações orientadas a servicos); Analise de sistemas; modelagem; Métricas; orientação à objetos; UML; testes e muito mais. **Assine já!**

DevMedia GROUP

Perguntas e Respostas

Acessando um compartilhamento do Windows sem ter permissão

Perguntas respondidas por Paulo Quicoli
pauloquicoli@devmedia.com.br

Recentemente recebi uma dica do colega e leitor Carlos (*carlos_am_2003@hotmail.com*) e fiz questão de publicar, pois achei bastante interessante.

Imagine a situação onde você possui um sistema instalado em algum cliente que possui uma rede *Windows*. Esse sistema precisa acessar uma pasta compartilhada que somente um determinado usuário possui permissão. Poderíamos utilizar o recurso de *Impersonate* para obter acesso, mas isso exigiria criar o usuário que tem permissão em todas as máquinas da rede. Parece uma ótima solução, mas seu cliente não quer fazer isso. Sabemos qual usuário possui acesso àquela pasta e não podemos criá-lo nas máquinas, como vamos resolver isso? Não é difícil, exigiu apenas uma consulta às *API's* do *Windows*.

A API que vamos utilizar é a *WNetAddConect2*. Ela realiza uma conexão com um recurso de rede. Por recurso de rede entenda pasta compartilhada e impressora compartilhada. Na **Listagem 1** temos a definição da chamada da função.

Listagem 1. Assinatura da função

```
DWORD WNetAddConnection2(
  LPNETRESOURCE lpNetResource,
  LPCTSTR lpPassword,
  LPCTSTR lpUsername,
  DWORD dwFlags
);
```

Vamos analisar os parâmetros requeridos. O primeiro parâmetro é uma estrutura do tipo *NETRESOURCE*. Essa estrutura contém informações sobre o recurso que desejamos acessar. O segundo e terceiro parâmetro formam a autenticação do usuário que possui a permissão de acesso ao recurso, ou seja, respectivamente senha e *login*, nesta ordem. O último parâmetro especifica algumas opções de conexão.

Utilizando a API no Delphi

A primeira coisa que temos que fazer é incluir na seção *Uses* a *unit ShellAPI*. Digamos agora que ao clique de um botão você quer acessar o compartilhamento *\\servidor\Documentos* que somente o usuário *Admin*, senha *TesteDelphi* tem acesso. Na **Listagem 2** vemos como isso fica.

Listagem 2. Acessando um recurso de rede

```
procedure TForm1.Button1Click(Sender: TObject);
var
 NetResource: TNetResource;
 senha, login: string;
begin
  senha := 'TesteDelphi';
  login := 'Admin';
  NetResource.dwType := RESOURCETYPE_Disk;
  NetResource.lpRemoteName := PChar('\\
   servidor\Documentos');
  netResource.lpLocalName := '';
  netResource.lpProvider  := '';
  WNetAddConnection2(NetResource,
   Pchar(senha), PChar(login), Connect_
    Update_Recent);
  CopyFile('c:\teste.txt','\\servidor\
   Documentos\teste.txt', false);
end;
```

Veja que temos uma variável do tipo *TNetResource*. O *Delphi* já possui esse tipo que na realidade mapeia para o tipo que a API necessita. Esse mapeamento é feito na *unit Windows*, que é incluída automaticamente em todo formulário *Delphi*. Nessa variável informamos o tipo de recurso que vamos acessar. No nosso exemplo vamos acessar uma pasta compartilhada, então especificamos *RESOURCETYPE_Disk* como tipo de recurso. Precisamos também informar onde ele está localizado. Na propriedade *lpRemoteName* informamos então o caminho do compartilhamento.

Caso fosse necessário mapear essa pasta para alguma unidade, na propriedade *lpLocalName* poderíamos definir isso. Quanto à última propriedade, *lpProvider*, deixamos vazio para que o sistema operacional identifique por si só qual a melhor forma de acesso. E para finalmente chamar a função passamos então a variável *NetResource*, que, como já mencionado contém as especificações do recurso que vamos acessar, a senha e *login* do usuário que tem permissão e por último, dizemos que desejamos adicionar essa conexão na lista das conexões recentes que o *Windows* mantém. Só assim vamos conseguir copiar o arquivo *Teste.txt* para a pasta compartilhada. ●

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Você não está mais sozinho.

A partir de agora você pode contar com a ajuda da

Chegou a Consultoria On-line DevMedia

Consultoria Técnica + Professor Virtual + Certificação

Mais Informações:
www.devmedia.com.br/consultoria_online 21 3382-5025

DevMedia em seus projetos e estudos.

A DevMedia possui um numeroso time de autores, editores e professores que juntos produzem o material que você está acostumado a encontrar em nosso site e revistas. E são exatamente esses mesmos profissionais que estarão a sua disposição para tirar suas dúvidas e ajudá-lo em seus projetos e estudos. Através de uma plataforma 100% web a Consultoria DevMedia garante sigilo absoluto, eficiência e rapidez em todas as respostas. Finalmente você terá ao seu alcance uma consultoria de qualidade por um preço muito acessível. Consulte nossos planos.

Mais um serviço DevMedia group

Nesta seção você encontra artigos intermediários sobre Delphi Win 32 e Delphi for .NET

Quick Update

Instalando componentes mais facilmente no Delphi

De que se trata o artigo

Definição, explicação e planejamento de recursos para a realização de um backup de um banco de dados em fita.

Para que serve

Proporciona uma explicação detalhada dos recursos envolvidos para a realização de um backup de um banco de dados em fita. Também pode ser utilizado como base para a criação de uma estratégia de backup de acordo com as características do banco de dados em questão.

Em que situação o tema é útil

O backup em fita é utilizado sempre que não há possibilidade de armazenar o backup de um banco de dados diretamente em disco rígido. As estratégias sugeridas no artigo também são úteis para DBAs que não conhecem os detalhes envolvidos no backup e precisam de um ponto de partida para montar sua própria estratégia de backup.

Adriano Santos
falecom@adrianosantos.pro.br
É desenvolvedor Delphi desde 1998, professor e programador PHP, bacharel em Comunicação Social pela Universidade Cruzeiro do Sul, SP. É Editor Chefe das revistas ClubeDelphi e WebMobile. Gerente de TI na SoftPark, parceira Borland, membro fundador do DUG-SP (www.dug-sp.com) e mantenedor do blog Delphi to Delphi (www.delphitodelphi.blogspot.com) com dicas, informações e tudo sobre desenvolvimento Delphi.

Olá, nessa edição o mérito todo vai para o colega e leitor Sérgio Guedes (*borlandlinuxer@gmail.com*). Ele nos mandou um artigo interessante sobre uma ferramenta chamada *DelphiPI*. O principal objetivo dessa ferramenta é facilitar a instalação de componentes no Delphi. Para quem está pouco acostumado ou simplesmente utiliza um bom volume de componentes em sua aplicação, o *DelphiPI* vem de encontro a essas necessidades. Seu código-fonte é open source, ou seja, código aberto. É possível efetuar o download dele no site *delphipi.googlecode.com/*. Vamos ver como a ferramenta funciona.

Baixe a versão 0.40 do site indicado em um local a sua escolha. Após efetuado o download execute a aplicação que solicitar a preferência de linguagem, confirme e aguarde o início da instalação. Prossiga confirmando cada tela de instalação, ao final marque a opção *Launch DelphiPI* e finalize.

Para demonstrar o *DelphiPI* vamos instalar um componente qualquer em nossa IDE. No meu caso vou utilizar

comum para validação de *CGC/CPF*.

Ao abrirmos o *DelphiPI*, notamos uma tela bastante simples como pode ser visto na **Figura 1**.

Figura 1. Tela principal do DelphiPI

O próximo passo é informar o caminho do arquivo do componente que se deseja instalar através do campo *Base Folder* no grupo *Select Base Folder contains both Package and Source Files*. O campo *Base Folder* é o campo onde informamos a base dos arquivos de um pacote, por exemplo: existem componente na internet que possuem uma série de arquivos *.pas* associados a ele, e por isso diversas pastas fazem parte de seu pacote.

Em seguida, precisamos informar o *Package File Pattern* do componente, ou seja, a extensão do arquivo que contém nosso componente, por padrão *.dpk*. Clique em *Next* para continuar.

Na próxima etapa, é necessário informarmos em qual Delphi queremos fazer a instalação e quais pastas serão gravados os arquivos do componente após a compilação. Note que é possível instalarmos em *Delphi 7*, *BDS 2006* e *RAD Studio 2007* (**Figura 2**). Novamente clique em *Next*.

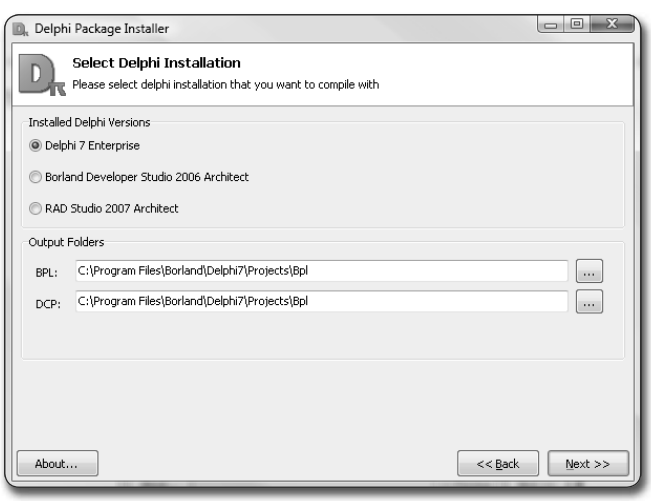


Figura 2. Escolhendo o Delphi

Automaticamente o *DelphiPI* localiza na pasta informada na primeira tela, que há um pacote de instalação no diretório e o marca como sendo um dos pacotes a instalar. Se tivéssemos mais de um *.dpk* na pastas, os mesmos seriam listados (**Figura 3**). Agora basta clicar em *Compile* e aguardar.

Depois de finalizada a instalação, basta abrir o Delphi que foi escolhido para tal instalação e notar que o componente foi instalado corretamente (**Figura 4**).

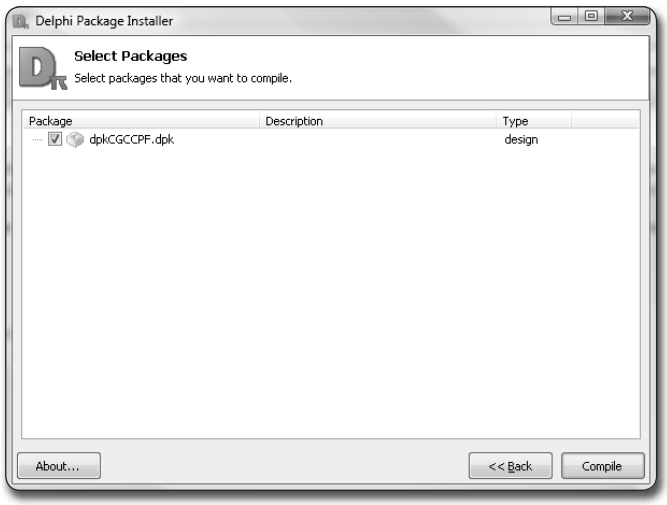


Figura 3. Lista de pacotes encontrados pelo DelphiPI

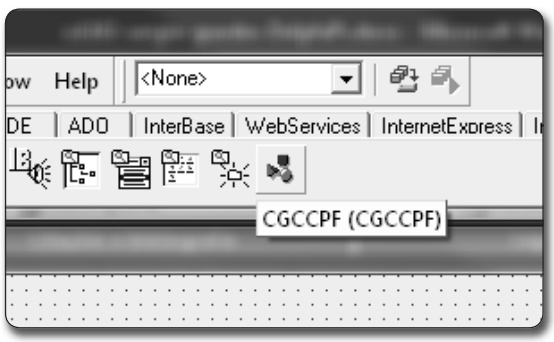


Figura 4. Componente instalado com sucesso

Com certeza o *DelphiPI* é uma ferramenta fundamental para agilizar o processo de instalação de componentes. Após receber a dica do leitor, fiz questão de testar diversos pacotes de instalação em diferentes versões do Delphi e realmente funciona muito bem o programa. Note também que a versão 0.40, mencionada nesse artigo, é compatível com a versão 2009 do Delphi. ●

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Nesta seção você encontra artigos intermediários sobre Delphi Win 32 e Delphi for .NET

Sistemas DataSnap com SOAP

Crie sistemas integrados usando SOAP – Parte I

Nesse artigo veremos

- Implementação de uma estrutura SOAP e DataSnap para agilizar processos entre empresa e fornecedor;
- Apresentação de um cenário onde a mesma estrutura pode ser explorada;
- Levantamento de conceitos para entendimento da tecnologia;
- Estruturação do banco de dados;
- Demonstração de um ambiente baseado em camadas, utilizando o método de comunicação SOAP para trafegar os dados via protocolo HTTP.

Qual a finalidade

Desenvolver algo prático, que possa ser usado no dia-a-dia, aplicando os mesmos métodos em diversos sistemas.

Quais situações utilizam esse recurso?

Todo e qualquer sistema que necessita de visões on-line do cenário, podem ser construídos utilizando os conceitos que aprenderemos nesse artigo.

Maikel Marcelo Scheid

maikelscheid@gmail.com

É técnico em Informática com ênfase em Análise e Programação de Sistemas. Atua na área de Desenvolvimento de Softwares em Delphi para plataforma Win32 e .NET com banco de dados Firebird e MSSQL.

Edinei Daniel Steffen

edineidaniel@gmail.com

É técnico e Bacharel em Sistemas de Informação, Analista e Desenvolvedor de Sistemas. Atua na área de Desenvolvimento de Softwares em Delphi para plataforma Win32 e .NET com banco de dados Firebird e MSSQL.

Conectividade, acessibilidade e agilidade são palavras que podem definir o novo modismo no que diz respeito ao desenvolvimento de softwares para controles industriais, tais como: de produção e PCP (*Programa de Controle de Produção*).

Nesse artigo iremos demonstrar de forma prática o uso da tecnologia *DataSnap* trabalhando com o protocolo *SOAP* (*Simple Object Access Protocol*). Ambas tecnologias estão disponíveis no *Delphi* e podem ser largamente utilizadas para os mais variados fins. O *DataSnap* é um conjunto de controles e conceitos que facilitam o desenvolvimento em camadas.

Resumo do DevMan

O desenvolvimento de sistemas em n camadas tem se popupalirazado no cenário de TI. Fazer com que uma aplicação possa ser gerenciada a distância via Web não é uma tarefa simples, porém o que veremos é a construção de uma solução completa utilizando o protocolo SOAP e o conceito DataSnap. Veremos que é perfeitamente possível colocar tudo em prática em apenas alguns passos.

A estrutura que será apresentada dará um enfoque exclusivo no uso da tecnologia SOAP e DataSnap para a integração de sistemas entre empresas, tratando do funcionamento e regras de negócio com algumas preocupações centralizadas de segurança.

Com essa facilidade, criar um sistema que possa ser acessado de qualquer parte do mundo acaba sendo uma das principais vantagens do *DataSnap*. Outra grande vantagem é, certamente, a velocidade tanto do desenvolvimento quanto do acesso aos dados. E velocidade conta bastante quando falamos em *Web* e sistemas integrados.

Sendo assim, criaremos uma aplicação que irá simular uma automação avançada entre indústrias do setor metal-mecânico e empresas fornecedoras de matérias-primas. A aplicação será desenvolvida em duas etapas, onde, nesse primeiro artigo, veremos desde o entendimento do problema até a criação e codificação da camada *Server*.

No segundo e último artigo da série, veremos a conexão do servidor por meio de duas aplicações cliente. Umas delas responsável pelos cadastros e gerenciamento do sistema e a outra que cuidará somente dos pedidos dos fornecedores. Faremos uma breve análise sobre o problema, passando ao levantamento de conceitos a serem utilizados no desenvolvimento, bem como proposição do banco de dados e o desenvolvimento da aplicação.

Entendimento do Problema

Para quem não sabe, uma linha de produção trabalha diariamente correndo alguns riscos, como por exemplo: falta de matéria-prima. Esses riscos certamente influenciam no faturamento da empresa. Quem controla, normalmente, os materiais e matérias-prima são gerentes de produção que tem que como principal função: organizar, cuidar e garantir a máxima eficiência de uma linha.

Um dos pontos-chave, para que uma linha de produção não sofra uma eventual parada por falta de algum tipo de material, gira em torno do controle e dos pedidos de compras. A principal regra para isso é justamente o controle do estoque mínimo de cada produto necessário. Tendo em vista essa situação, desenvolveremos um sistema capaz de detectar o limite mínimo de estoque e, automaticamente, emitir pedidos de compras avisando então os gerentes de produção.

Deste modo definimos que o sistema será desenvolvido em três camadas: *Camada do Banco de Dados*, *Camada do Servidor* (onde estarão as regras de negócio) e as *Camadas Cliente* que serão acessadas tanto para a empresa como para os fornecedores.

Client/Server x Três Camadas

Mas o que realmente muda em uma aplicação três ou *n* camadas para o método convencional (*Client/Server*)? Veja a análise dos dois casos.

- No método *duas camadas* (*Client/Server*): quando trabalhamos com eventos de componentes visuais, é fácil dispersarmos algumas regras como as consistências de campos. Isso pode acarretar em dores de cabeça no momento de uma manutenção, bem como alguns problemas ligados a distribuição do sistema. Se uma regra for mudada em um formulário, todo o sistema precisa ser *recompilado* e distribuído novamente.
- Analisando o método de três ou *n* camadas, podemos perceber que todas as regras de negócio localizam-se na camada de meio da aplicação, ou seja, a camada que fica entre o *banco de dados* e *cliente* junto ao servidor de aplicação.

Nota

DataSnap é uma coletânea de tecnologias que funcionam em conjunto e tem o objetivo de facilitar o desenvolvimento de aplicativos distribuídos.

Outra definição que também cabe citar aqui é de que *SOAP* foi definido em *XML*, para tanto a comunicação (remota) entre as camadas é baseada em *XML* e o transporte da mensagem é feita através do *HTTP* (*protocolo de transporte de dados na Internet*) permitindo que a comunicação aconteça sem bloqueios entre as camadas.

Banco de Dados

A **Figura 1** exibe de uma forma bem simples como se dará a organização e armazenamento de dados. Nele serão armazenadas as informações de produtos, fornecedores e pedidos com seus itens. Lembrando que os pedidos serão gerados automaticamente assim que os mesmos chegarem a seus limites mínimos de estoque para seus respectivos fornecedores.

O banco de dados definido para o desenvolvimento da aplicação foi o *Firebird 1.5* e conforme a **Figura 1** é proposto o *script* de criação do banco de dados demonstrado na **Listagem 1**. Utilize um gerenciador de banco de dados, como o *IBExpert* por exemplo, ou qualquer outro para executar o *script*.

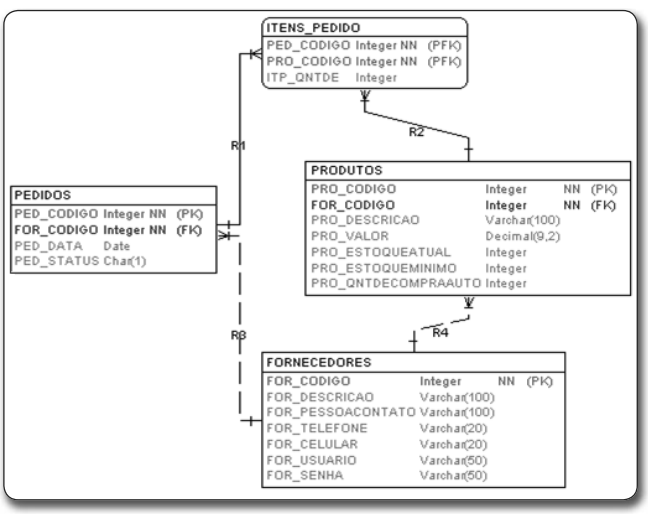


Figura 1. Estrutura proposta para o Banco de Dados

Listagem 1. Script para criação da base de dados

```
SET SQL DIALECT 3;
SET NAMES WIN1252;

SET CLIENTLIB '<caminho>\fbclient.dll';

CREATE DATABASE '<caminho>\BD_PEDIDOS.FDB'
USER 'SYSDBA' PASSWORD 'masterkey'
PAGE_SIZE 4096
DEFAULT CHARACTER SET WIN1252;

CREATE GENERATOR GEN_FORNECEDORES_ID;
SET GENERATOR GEN_FORNECEDORES_ID TO 0;

CREATE GENERATOR GEN_PEDIDOS_ID;
SET GENERATOR GEN_PEDIDOS_ID TO 0;

CREATE GENERATOR GEN_PRODUTOS_ID;
SET GENERATOR GEN_PRODUTOS_ID TO 0;

CREATE TABLE FORNECEDORES (
    FOR_CODIGO          INTEGER NOT NULL,
    FOR_DESCRICAO       VARCHAR(100),
    FOR_PESSOACONTATO   VARCHAR(100),
    FOR_TELEFONE        VARCHAR(20),
    FOR_CELULAR         VARCHAR(20),
    FOR_USUARIO         VARCHAR(50),
    FOR_SENHA           VARCHAR(50)
);

CREATE TABLE ITENS_PEDIDO (
    PED_CODIGO  INTEGER NOT NULL,
    PRO_CODIGO  INTEGER NOT NULL,
    ITP_QNTDE   INTEGER
);

CREATE TABLE PEDIDOS (
    PED_CODIGO  INTEGER NOT NULL,
    FOR_CODIGO  INTEGER NOT NULL,
    PED_DATA    DATE,
    PED_STATUS  CHAR(1)
);

CREATE TABLE PRODUTOS (
    PRO_CODIGO            INTEGER NOT NULL,
    FOR_CODIGO            INTEGER NOT NULL,
    PRO_DESCRICAO         VARCHAR(100),
    PRO_VALOR             NUMERIC(9,2),
    PRO_ESTOQUEATUAL      INTEGER,
    PRO_ESTOQUEMINIMO     INTEGER,
    PRO_QNTDECOMPRAAUTO   INTEGER
);

ALTER TABLE FORNECEDORES ADD PRIMARY KEY (FOR_CODIGO);
ALTER TABLE ITENS_PEDIDO ADD PRIMARY KEY (PED_CODIGO, PRO_CODIGO);
ALTER TABLE PEDIDOS ADD PRIMARY KEY (PED_CODIGO);
ALTER TABLE PRODUTOS ADD PRIMARY KEY (PRO_CODIGO);

SET TERM ^ ;

CREATE TRIGGER FORNECEDORES_BI FOR FORNECEDORES
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  IF (NEW.FOR_CODIGO IS NULL) THEN
    NEW.FOR_CODIGO = GEN_ID(GEN_FORNECEDORES_ID,1);
END
^

CREATE TRIGGER PEDIDOS_BI FOR PEDIDOS
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  IF (NEW.PED_CODIGO IS NULL) THEN
    NEW.PED_CODIGO = GEN_ID(GEN_PEDIDOS_ID,1);
END
^

CREATE TRIGGER PRODUTOS_BI FOR PRODUTOS
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  IF (NEW.PRO_CODIGO IS NULL) THEN
    NEW.PRO_CODIGO = GEN_ID(GEN_PRODUTOS_ID,1);
END
^

SET TERM ; ^
```

Entendendo o funcionamento

O funcionamento da aplicação se dará baseado na **Figura 2**, simulando uma máquina cliente dentro da empresa que é responsável pela alimentação dos dados no sistema e realizar o controle de saída de materiais. Esta máquina pode estar na rede local da empresa ou em qualquer outro lugar na Internet. Já na máquina cliente do fornecedor, mediante um usuário e senha de identificação, apenas será disponibilizada uma interface para visualização dos pedidos que serão emitidos automaticamente pelo sistema central ou camada central onde se encontram as regras de negócio. Somente esta camada fará uma conexão com o banco de dados e disponibilizará as informações em um formato *XML* para as outras camadas.

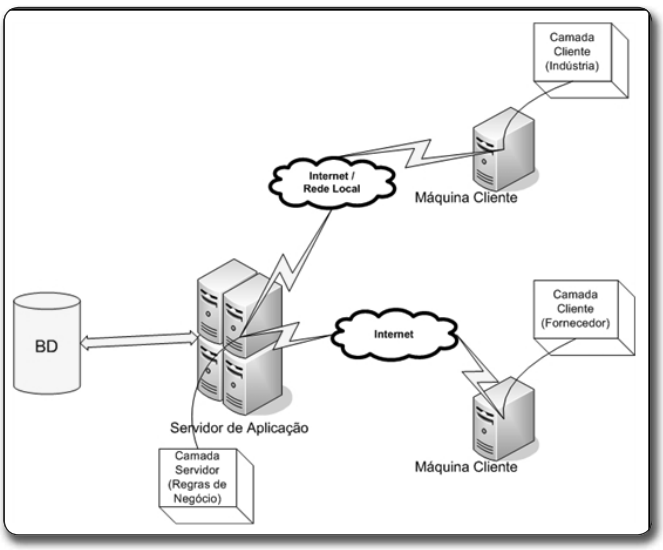


Figura 2. Visão gráfica do funcionamento da aplicação

Regras do Servidor

Como dito anteriormente, o nosso sistema será composto por algumas aplicações. Uma delas é nosso servidor de aplicação, que funcionará no servidor central da empresa. Para que possamos desenvolvê-lo, precisamos estabelecer algumas regras e então aplicá-las ao programa. Veja a seguir as principais regras de nosso sistema.

Nosso sistema deverá:

- Disponibilizar as informações para que o cliente possa realizar as manutenções de cadastros;
- Possibilitar abertura de pedidos para fornecedores;
- Incluir produtos em uma ordem de pedidos não fechada;
- Disponibilizar dados de pedidos por fornecedor;
- Finalizar pedidos emitidos para um fornecedor;
- Atualizar estoque dos produtos;
- Identificar fornecedor pelo usuário e senha do banco de dados.

Perceba que nossas regras na verdade são tarefas que deverão ser realizadas pelo servidor de aplicação. Em outras palavras, ele servirá as aplicações cliente como um *executor de tarefas*. É ele quem receberá as ordens vindas de cada aplicação e executará no servidor. Essa é a essência básica de um servidor de aplicação.

Desenvolvimento da solução proposta

Após a montagem do cenário no qual será desenvolvida a solução, iniciaremos com a criação da camada de *negócios*, onde serão desenvolvidos todos os procedimentos e regras para o funcionamento das outras camadas. Para o desenvolvimento será utilizado o *CodeGear Rad Studio 2007 for Win32*, mas qualquer outra versão do Delphi poderá ser utilizada.

No *RAD 2007*, em *File>New>Other>Delphi Projects> WebServices>SOAP Server Application*. Após escolhida esta opção deverá ser definido o tipo de estrutura *Web* para o sistema, no caso, neste exemplo iremos utilizar *ISAPI/NSAPI Dynamic Link Library*, conforme ilustrado na **Figura 3**.

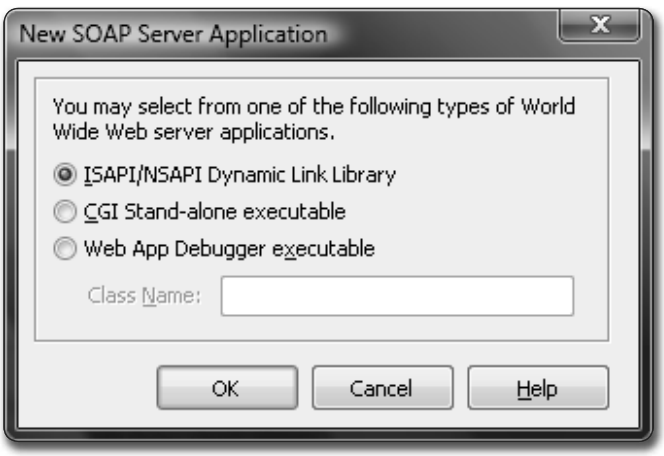


Figura 3. Criando a aplicação WebServices

Definido o tipo da aplicação, surgirá uma caixa de diálogo com a pergunta *Create Interface for SOAP module?*. Responda que não (*No*) para que não seja criada uma interface para métodos públicos do serviço *SOAP*. Faremos isso em outro momento.

Salve os arquivos da aplicação (*File>Save All*) deixando a *unit* principal como *uWMPedidos.pas* e o projeto como *PedidosSoapServer.dpr*. Após salvar o projeto, perceba que a Unit1, salva como uWMPedidos.pas, é na verdade um *WebModule*. Modifique sua propriedade *Name* para *WMPedidos* e deixe o restante da estrutura como foi criada pelo Delphi. Em seguida acesse menu *File>New>Other>Delphi Projects>WebServices>SOAP Server Data Module*. Na janela que aparece, dê o nome de *DMPedidos* ao *DataModule* e em seguida salve-o como *uDMPedidos.pas*.

A estrutura inicial da camada servidora está configurada, agora é necessário adicionar os componentes que farão a interação com o banco de dados. Todos os componentes que utilizaremos serão adicionados no *DMPedidos*.

Iniciando com o componente *SQLConnection* (*sqlConexao*) da paleta *dbExpress* para viabilizar a conexão com o banco de dados. Acesse sua propriedade *Params* configurando-a de acordo com informações da **Figura 4**. Aproveite e configure a propriedade *Login Prompt* para *False*.

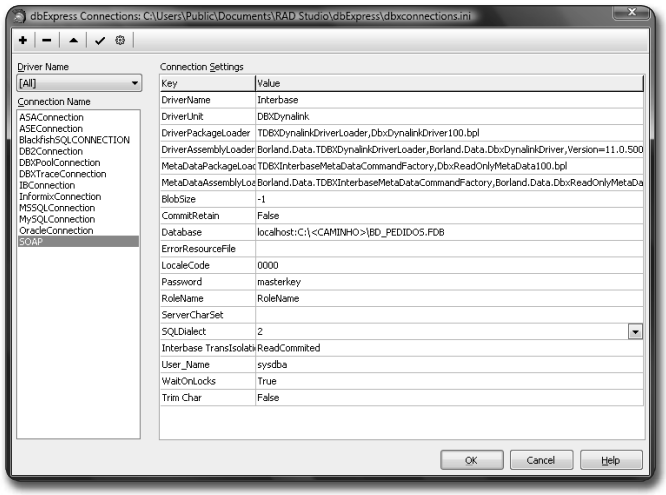


Figura 4. Configuração do componente de conexão com o banco de dados

Realizada a configuração do componente de conexão com o banco, passaremos a utilizar os componentes para acessar e manipular os dados das diversas tabelas do banco de dados.

Os componentes indicados para este fim serão o *SQLQuery e DataSetProvider* das paletas *dbExpress* e *DataAccess*, respectivamente.

De imediato adicione cinco componentes *SQLQuery* (*sqlProdutos*, *sqlFornecedores*, *sqlPedidos*, *sqlItens_Pedido* e *sqlPedidosPorFornecedor*) e na propriedade *SQLConnection* de cada componente, selecione o componente de conexão e em suas propriedades *SQL* adicione os comandos *SQL`s*, na ordem em que aparecem na **Listagem 2.**

Note que alguns comandos *SQL* possuem um parâmetro para ser informado antes da abertura ou execução do comando no banco de dados. É muito importante fazer o uso de parâmetros para resgatar os dados de uma tabela principalmente quando se trata de uma estrutura de comunicação *HTTP*, fazendo com que não haja a necessidade de toda a vez que for solicitado uma consulta se retornar todos os dados da tabela. A presença da cláusula *WHERE* na instrução *SQL* se torna indispensável para este tipo aplicação.

Devido à presença de um parâmetro na *SQL*, devemos também configurar o tipo de dado que deve ser informado nesse parâmetro. Clicando na propriedade *Params* e selecionando o parâmetro desejado, altere a propriedade *DataType* para *ftString* para valores do tipo texto ou *ftInteger* para tipos de dados inteiros.

Ative os componentes marcando como *True* a propriedade *Active* de cada componente. Abra o *FieldsEditor* de cada *Query* e adicione todos os campos clicando com botão direito do mouse e escolhendo a opçãp *Add all Fields*.

Algo que também deve ser observado nesta configuração são os *ProviderFlags*. Essa propriedade permite que possamos configurar quais os campos fazem parte ou não da chave primária da tabela além, é claro, de informar ao *DataSetProvider* o que fazer com cada campo enviado ou recebido pela *Query*. Os campos que fazem parte da chave primária da tabela devem ter a seguinte configuração na propriedade *ProviderFlags*:

Listagem 2. Comandos SQL

```
[sqlProdutos]
SELECT
  PRODUTOS.PRO_CODIGO, PRODUTOS.FOR_CODIGO,
  PRODUTOS.PRO_DESCRICAO, PRODUTOS.PRO_VALOR, PRODUTOS.PRO_ESTOQUEATUAL,
  PRODUTOS.PRO_ESTOQUEMINIMO, PRODUTOS.PRO_QNTDECOMPRAAUTO
FROM
  PRODUTOS
WHERE
  (PRODUTOS.PRO_DESCRICAO LIKE :PDESCRICAO)
ORDER BY
  PRODUTOS.PRO_DESCRICAO

[sqlFornecedores]
SELECT
  FORNECEDORES.FOR_CODIGO, FORNECEDORES.FOR_DESCRICAO,
  FORNECEDORES.FOR_PESSOACONTATO, FORNECEDORES.FOR_TELEFONE,
  FORNECEDORES.FOR_CELULAR, FORNECEDORES.FOR_USUARIO,
  FORNECEDORES.FOR_SENHA
FROM
  FORNECEDORES
WHERE
  (FORNECEDORES.FOR_DESCRICAO LIKE :PFOR_DESCRICAO)
ORDER BY
  FORNECEDORES.FOR_DESCRICAO

[sqlPedidos]
SELECT
  PEDIDOS.PED_CODIGO, PEDIDOS.FOR_CODIGO, PEDIDOS.PED_DATA,
  PEDIDOS.PED_STATUS
FROM
  PEDIDOS
ORDER BY
  PEDIDOS.PED_DATA

[sqlItens_Pedido]
SELECT
  ITENS_PEDIDO.PED_CODIGO, ITENS_PEDIDO.PRO_CODIGO,
  ITENS_PEDIDO.ITP_QNTDE, PRODUTOS.PRO_DESCRICAO
FROM
  ITENS_PEDIDO
  INNER JOIN PRODUTOS ON
  (ITENS_PEDIDO.PRO_CODIGO = PRODUTOS.PRO_CODIGO)
WHERE
  (ITENS_PEDIDO.PED_CODIGO = :PPED_CODIGO)
ORDER BY
  PRODUTOS.PRO_DESCRICAO

[sqlPedidosPorFornecedor]
SELECT
  PEDIDOS.PED_CODIGO, PEDIDOS.FOR_CODIGO, PEDIDOS.PED_DATA,
  PEDIDOS.PED_STATUS
FROM
  PEDIDOS
WHERE
  (PEDIDOS.FOR_CODIGO = :PFOR_CODIGO) AND (PEDIDOS.PED_STATUS = 'A')
ORDER BY
  PEDIDOS.PED_DATA
```

[pfInUpdate, prInWhere, pfInKey]. E para os demais campos deve somente estar ativada a *flag*: *[pfInUpdate]*.

Necessitamos também de cinco componentes *DataSetProvider (dspProdutos, dspFornecedores, dspPedidos, dspItens_Pedido, dspPedidosPorFornecedor)* da paleta *DataAccess* relacionados, através da propriedade *DataSet*, com os respectivos componentes *SQLQuery* (*sqlProdutos, sqlFornecedores, sqlPedidos, sqlItens_Pedido, sqlPedidosPorFornecedor*) e na propriedade *UpdateMode* selecione *upWhereKeyOnly*.

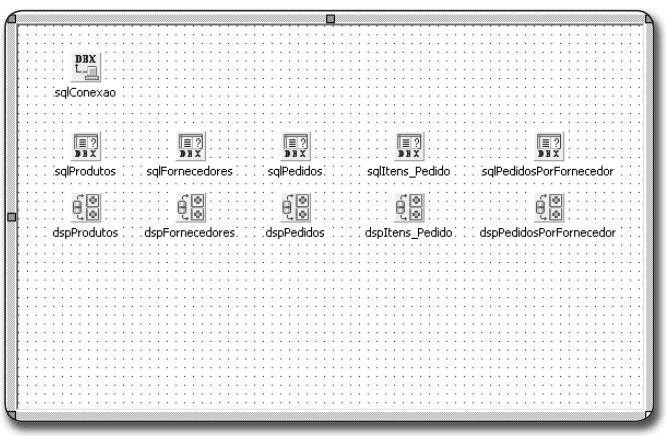


Figura 5. Relação de componentes necessários

Configurados os componentes de acesso e manipulação dos dados do sistema no servidor da nossa aplicação, precisamos agora criar a interface que fará a comunicação com as aplicações cliente e que farão acesso através do protocolo *SOAP*. Na interface do servidor serão criadas as funções e métodos que cuidarão das regras de negócio, ou seja, procedimentos de cadastros, pesquisa, verificações, validações, segurança, entre outros.

No menu *File>New>Other>Delphi Projects>WebServices>SOAP Server Interface* (**Figura 6**), que ao ser executado irá solicitar que o nome do serviço de interface seja informado. Chame o serviço de *funcoes* e em seguida desmarque o item *Generate comments* e confirme o diálogo criando assim dois novos arquivos de código no servidor (*funcoesImpl.pas* e *funcoesIntf.pas*). Sendo eles a interface visível através de conexões realizadas e outro com as implementações de código que ficarão centralizadas nesta camada *Server*.

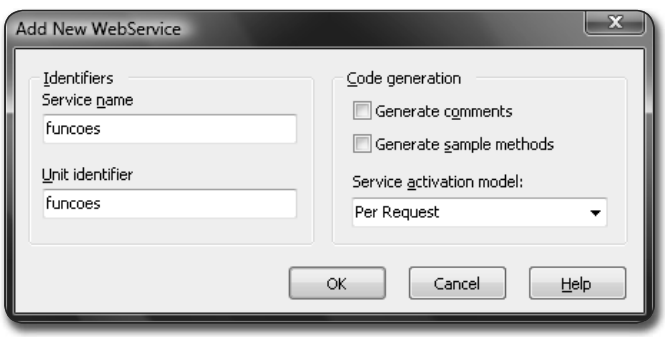


Figura 6. Criando a interface da camada *server*

A *unit funcoesIntf.pas* será responsável por exportar, ou seja, disponibilizar e tornar visível à *Web* as funções a serem invocadas pela camada servidora. Nessa *unit* faremos somente as declarações de cada função, pois a implementação de cada uma será feita em outra *unit*. Ao final de cada *function/procedure* declarada iremos adicionar *stdcall* indicando que os parâmetros serão exportados.

Declare as funções e procedimentos de acordo com o código da **Listagem 3**, onde encontram-se listados todas as *functions* e *procedures* necessárias para a manipulação da camada servidora através da conexão de camadas clientes.

Listagem 3. Implementando os códigos do server

```
unit funcoesIntf;

interface

uses
  InvokeRegistry, Types, XSBuiltIns;

type
  Ifuncoes = interface(IInvokable)
  ['{4389A2ED-61D6-4A59-AC50-2F9DFDC7CDC9}']
  function MontaPedidos(): String; stdcall;
  function AtualizarEstoqueProduto(CodigoProduto, Qntde, Operacao:
    String): Boolean; stdcall;
  function AlterarStatusPedido(CodigoPedido, Status: String):
    string; stdcall;
  function IdentificaFornecedor(Usuario, Senha: String;
    var Codigo, Nome: String): boolean; stdcall;
  end;

implementation

initialization
  InvRegistry.RegisterInterface(TypeInfo(Ifuncoes));
end.
```

Observe que estamos realizando a exportação de quatro funções responsáveis pela montagem da lista de novos pedidos, atualização de quantidades de produtos solicitadas no estoque, alteração de *status* de pedidos já existentes e também a identificação do fornecedor solicitante através de usuário e senha fornecidos. Os parâmetros para estas funções serão trafegados de forma criptografada, dificultando a intervenção de invasores na rede e captura ou alteração das informações.

Partindo agora para a implementação das funções na *unit funcoesImpl.pas*. Temos a classe *TInvokableClass* que é a responsável pela estrutura de todas as classes que implementam interfaces *IInvokable*. Esta classe é semelhante à classe *TInterfacedObject* usada em aplicações *COM*. Nesta *unit* é que faremos todas as regras da nossa camada *Server*, ou seja, vamos implementar as funções a serem exportadas e além desta, criar outras para que façam um gerenciamento silencioso de todos os processos que são executados.

Faça a declaração das funções e procedimentos conforme a **Listagem 4**. Perceba que na linha 8 fiz a declaração de uma constante chamada *SegKey*. Essa constante será nossa chave de segurança, como um *Token*. Isso evita que alguém acesse o banco de dados diretamente por outros meios. Para que o Delphi crie a estrutura de cada *function/procedure* pressione simultaneamente *Ctrl + Shift + C*.

Antes de detalharmos cada função, vejamos resumidamente a função que cada uma desempenhará no sistema. Na sessão *private* temos declaradas as funções/procedimentos: *MontaPedidoFornecedor, AdicionaProdutoPedido, VerificaPedidoProduto, VerificaPedidoAbertoFornecedor* e *Seguranca*. Veja a seguir a descrição resumida delas:

- *MontaPedidoFornecedor*: procedimento auxiliar utilizado na montagem de pedidos para fornecedores, fazendo que com seja possível abrir pedidos por fornecedor de acordo com o valor que é atribuído ao parâmetro;
- *AdicionaProdutoPedido*: procedimento responsável por adicionar um produto ao pedido criado ou em edição (aberto), sendo necessário informar o código do mesmo, o código do produto e a quantidade a ser solicitada;
- *VerificaPedidoProduto*: função responsável por verificar se um determinado produto já se encontra em algum pedido que está com seu *status Aberto*, retornando um valor *booleano (Verdadeiro/Falso)*;
- *VerificaPedidoAbertoFornecedor*: como o próprio nome já sugere, esta função é responsável pela verificação de um pedido que está *Aberto* para uma determinado fornecedor. Retorna ocorrências do tipo *True* ou *False* de acordo com o resultado da busca.
- *Seguranca*: função responsável pela criptografia e descriptografia das informações que serão trafegadas entre as aplicações clientes e servidor. A função deverá receber como parâmetros a ação *E=Criptografar, D=Descriptografar*, o texto a ser transformado e a chave que é utilizada na criptografia dos caracteres. Esse último parâmetro é justamente nossa constante *SegKey*.

Listagem 4. Declaração das funções do servidor

```
01 unit funcoesImpl;
02
03 interface
04
05 uses
06   InvokeRegistry, Types, XSBuiltIns, funcoesIntf, SQLExpr;
07
08 const SegKey = 'SOAP';
09
10 type
11   { Tfuncoes }
12   Tfuncoes = class(TInvokableClass, Ifuncoes)
13   private
14     procedure MontaPedidoFornecedor(Fornecedor: String);
15     procedure AdicionaProdutoPedido(Pedido, Produto, Qntde:
16       string);
17     function VerificaPedidoProduto(Fornecedor, Produto:
         String): boolean;
18     function VerificaPedidoAbertoFornecedor(Fornecedor:
         String): String;
19     function Seguranca(Acao, Texto, Chave: WideString):
         WideString;
21   public
22     function MontaPedidos(): string; stdcall;
23     function AtualizarEstoqueProduto(CodigoProduto, Qntde,
         Operacao:
24       string): Boolean; stdcall;
25     function AlterarStatusPedido(CodigoPedido, Status:
         string): string; stdcall;
26     function IdentificaFornecedor(Usuario, Senha: string; var
         Codigo, Nome:
27       string): boolean; stdcall;
28   end;
```

A sessão *public* possui as mesmas funções declaradas na *unit funcoesIntf.pas*. Isso acontece porque nossa classe atual é herdada de *Ifuncoes*, veja:

```
Tfuncoes = class(TInvokableClass, Ifuncoes)
```

A maior vantagem em se trabalhar com interfaces, é que elas funcionam como contratos nos obrigando a ter exatamente a mesma estrutura que a interface mãe. Em outras palavras, todas as funções e procedimentos declarados em *Ifuncoes* devem ser implementadas *Tfuncoes*, nossa classe atual. Isso facilita o trabalho e evita que cada desenvolvedor na equipe, crie suas próprias funções sem respeitar a hierarquia de ações.

Nós precisamos declarar então *MontaPedidos, AtualizaEstoqueProduto, AlterarStatusPedido* e *IdentificaFornecedor*. Veja a descrição resumida de cada função/procedimento:

- *MontaPedidos*: função responsável pela montagem dos pedidos com os produtos cujo seus estoques atuais estão iguais ou abaixo do estoque mínimo configurado no momento do cadastro do produto. Montando uma lista de produtos por fornecedor, o sistema realiza ainda verificações se já existe um pedido aberto para o fornecedor adicionando mais um produto à lista, caso contrário, um novo pedido é criado;
- *AtualizaEstoqueProduto*: função responsável pela atualização da quantidade em estoque do produto. É utilizada no momento de aumentar ou diminuir a quantidade, dependendo do valor informado na operação (*soma, subtração*), tendo como retorno um valor *verdadeiro* ou *falso* para identificar o sucesso ou não da operação;
- *AlterarStatusPedido*: função utilizada para a atualização do *status* do pedido. Cada fornecedor terá seu sistema *cliente* com a lista de pedidos que estejam abertos. Assim, a cada pedido que for encaminhado ao servidor, o *Status* desse pedido deve ser alterado para não correr o risco de um produto estar na lista e não ser enviado;
- *IdentificaFornecedor*: função responsável pela validação do *login* dos fornecedores junto a seus sistemas, retornando seu respectivo código e nome por parâmetro de retorno;

Codificando e entendendo o sistema

Aparentemente o nosso sistema parece ser bastante complexo, mas veremos nesse tópico que na verdade todo o processo de desenvolvimento com *SOAP* é bem simples e objetivo. No tópico anterior pressionamos Ctrl + Shift + C para que o Delphi nos criasse os cabeçalhos de cada função/procedimento. Vamos agora codificar cada um e entender seu funcionamento. Comecemos pelo procedimento *AdicionaProdutoPedido* (**Listagem 5**).

No início do procedimento (*Linhas 3* e *4*) declaramos duas variáveis que serão responsáveis por receberem os ponteiros de memória para o componente *SQLDataSet* e o *DataModule DMPedidos* que criaremos em *runtime*. Perceba que os criamos nas linhas 7 e 8. Em seguida atualizamos as propriedades do nosso *SQLDataSet* (*SQL*) utilizando um bloco *with..do* (*Linha 10*) para facilitar nosso trabalho de digitação dos procedimentos. Note que atualizamos a propriedade *SQLConnection* apontando-a para o *DataModule DMPedidos* e então modificamos sua propriedade *CommandText* incluindo uma instrução *SQL* que fará a inclusão (*INSERT*) dos itens de pedido na tabela de mesmo nome (*Linhas 14* e *15*). Na seqüência atualizamos os parâmetros *PED_CODIGO, PRO_CODIGO* e *ITP_QNTDE* passando o conteúdo das variáveis *Pedido, Produto* e *Qntde* recebidas por parâmetro pelo procedimento (*Linhas 16* a *18*). Por fim, chamamos o método *ExecSQL* (*Linha 19*) que faz a execução do comando *SQL* no servidor.

Todo esse procedimento é envolvido em um bloco *try..finally* (*Linhas 9, 21* e *25*). Isso evita que um erro ocorra e os objetos *SQL* e *DM* não sejam liberados da memória.

Nota do DevMan

É extremamente importante fazer o controle de transação em operações dessa natureza em toda e qualquer aplicação. Para fazer o controle de transação utilizando DBExpress de maneira mais simples, basta criarmos uma variável do tipo TransactionDesc e atribuí-la aos métodos StartTransaction, Commit e RollBack do SQLConnection. Além disso precisamos atribuir um ID a variável e informar qual o nível de isolamento da transação. Veja a seguir um exemplo:

```
procedure TfrmProcesso.btnGravarClick(Sender: TObject);
var
 Transacao: TTransactionDesc;
begin
 try
  Transacao.TransactionID := 1;
  Transacao.IsolationLevel := xilREADCOMMITTED;
  SQLConnection.StartTransaction(Transacao);
  { Procedimentos de gravação }
  SQLConnection.Commit(Transacao);
 except
  SQLConnection.Rollback();
 end;
end;
```

Perceba que isolamos o procedimento em um bloco try..except o que significa que caso haja um problema, toda a operação será cancelada pelo método RollBack.

Listagem 5. Procedimento AdicionaProdutoPedido

```
01 procedure Tfuncoes.AdicionaProdutoPedido(Pedido, Produto,
    Qntde: String);
02 var
03   sql: TSQLDataSet;
04   DM : TDMPedidos;
05 begin
06   try
07     sql := TSQLDataSet.Create(nil);
08     DM := TDMPedidos.Create(nil);
09     try
10       with sql do
11       begin
12         SQLConnection := DM.sqlConexao;
13         CommandText :=
14           'INSERT INTO ITENS_PEDIDO (PED_CODIGO,PRO_
               CODIGO,ITP_QNTDE) '+
15           ' VALUES (:PED_CODIGO,:PRO_CODIGO,:ITP_QNTDE) ';
16         ParamByName('PED_CODIGO').AsString := Pedido;
17         ParamByName('PRO_CODIGO').AsString := Produto;
18         ParamByName('ITP_QNTDE').AsString  := Qntde;
19         ExecSQL();
20       end;
21     finally
23       sql.free;
24       DM.free;
25     end;
26  except
27  end;
28 end;
```

Como vimos, não há segredos no procedimento *AdicionaProdutoPedido*. Agora, vejamos outro procedimento importante de nosso sistema. A *procedure AlterarStatusPedido* tem como principal função alterar o estado do pedido como seu próprio nome sugere. Logo de início podemos ver que o procedimento começa bem semelhante ao procedimento

anterior, ou seja, declarando as variáveis que serão usadas para instanciar um *SQLDataSet* e o *DMPedidos*. Saltamos então para as *linhas 14* até *21*. Nelas montamos uma instrução *SQL* para atualizar (UPDATE) o pedido em questão. Atualizamos os parâmetros *PPED_STATUS* e *PPED_CODIGO* usando o valor que veio nos parâmetros da *function* e por fim chamamos o método *ExecSQL* para executar o comando. A novidade aqui é que criamos uma *function* para esse processo, assim podemos receber o estado final da operação, ou seja, se foi ou não bem sucedido (*Linha 21*) (**Listagem 6**).

Listagem 6. Procedimento AlterarStatusPedido

```
01 function Tfuncoes.AlterarStatusPedido(CodigoPedido, Status:
    String): string;
02 var
03   sql: TSQLDataSet;
04   DM : TDMPedidos;
05 begin
06   result := '';
07   try
08     sql := TSQLDataSet.Create(nil);
09     DM := TDMPedidos.Create(nil);
10     try
11       with sql do
12       begin
13         SQLConnection := DM.sqlConexao;
14         CommandText :=
15           ' UPDATE PEDIDOS '+
16           ' SET PED_STATUS = :PPED_STATUS  '+
17           ' WHERE (PED_CODIGO = :PPED_CODIGO) ';
18         ParamByName('PPED_STATUS').AsString :=
            Seguranca('D',Status,SegKey);
19         ParamByName('PPED_CODIGO').AsString := Seguranca('D',
            CodigoPedido,SegKey);
20         ExecSQL();
21         Result := 'Pedido atualizado com sucesso!';
22       end;
23     finally
24       sql.free;
25       DM.free;
26     end;
27   except
28     Result := 'Problemas com a atualização do Pedido!';
29   end;
30 end;
```

Outra função importante de nosso sistema tem o nome de *AtualizarEstoqueProduto*. Após praticamente todas as operações no banco de dados, o estoque deve ser consultado e atualizado para evitar surpresas no futuro. Essa função tem exatamente essa responsabilidade. Veja sua implementação na **Listagem 7**

Vamos novamente pular diretamente para as linhas 15 a 23 já que o início de nossa função já é conhecido. Aqui estamos montando uma instrução SQL para atualizar o banco de dados com a quantidade de estoque restante. Repare que informamos na cláusula *Where* de nossa instrução o código do produto (*PRO_CODIGO*) e o estoque atual (*PRO_ESTOQUEATUAL*). O último parâmetro como sempre é a chave de validação, o *SegKey*. Então executamos o procedimento e resultamos em *True* ou *False*.

A *procedure MontaPedidoFornecedor* tem o mesmo princípio das demais funções que já vimos. Sua responsabilidade principal é incluir o pedido de compras no banco de dados, por isso veja sua instrução SQL nas linhas *14* e *15* da **Listagem 8**. O resumo é que apenas informamos o código do fornecedor e o pedido é incluído no sistema. O segundo e o terceiro parâmetro são a data e o estado do pedido, respectivamente.

Listagem 7. Código da função de atualização do estoque dos produtos

```
01 function Tfuncoes.AtualizarEstoqueProduto(CodigoProduto, Qntde,
02   Operacao: String): Boolean;
03 var
04   sql: TSQLDataSet;
05   DM : TDMPedidos;
06 begin
07   result := false;
08   try
09     sql := TSQLDataSet.Create(nil);
10     DM := TDMPedidos.Create(nil);
11     try
12       with sql do
13       begin
14         SQLConnection := DM.sqlConexao;
15         CommandText :=
16           'UPDATE PRODUTOS '+
17           ' SET PRO_ESTOQUEATUAL = PRO_ESTOQUEATUAL ' + Operacao +
18           ' :PRO_ESTOQUEATUAL  '+
19           ' WHERE (PRO_CODIGO = :PRO_CODIGO) ';
20         ParamByName('PRO_ESTOQUEATUAL').AsString :=
            Seguranca('D',Qntde,SegKey);
21         ParamByName('PRO_CODIGO').AsString :=
22           Seguranca('D',CodigoProduto,SegKey);
23         ExecSQL();
24         result := true;
25       end;
26     finally
27       sql.free;
28       DM.free;
29     end;
30   except
31     result := false;
32   end;
33 end;
```

Listagem 8. Procedimento de inclusão de pedidos

```
01 procedure Tfuncoes.MontaPedidoFornecedor(Fornecedor: String);
02 var
03   sql: TSQLDataSet;
04   DM : TDMPedidos;
05 begin
06   try
07     sql := TSQLDataSet.Create(nil);
08     DM := TDMPedidos.Create(nil);
09     try
10       with sql do
11       begin
12         SQLConnection := DM.sqlConexao;
13         CommandText :=
14           'INSERT INTO PEDIDOS (FOR_CODIGO,PED_DATA,PED_STATUS) '+
15           'VALUES (:FOR_CODIGO,''TODAY'',''A'')';
16         Params[0].AsString := Fornecedor;
17         ExecSQL();
18       end;
19     finally
20       sql.free;
21       DM.free;
22     end;
23   except
24   end;
25 end;
```

A montagem do pedido propriamente dito é um pouco mais extensa, mas não é complexa. Veja na **Listagem 9** toda sua implementação. Nas *linhas 17* e *18* está a instrução *SQL* responsável por resgatar os dados produtos. Logo em seguida verificamos se a consulta retornou dados (*IsEmpty*) o que significa que existe o produto consultado. Na seqüência envolvemos a operação em um laço *while..do* para que possamos incluir cada produto ao pedido. Perceba que chamamos as funções *VerificaPedidoAbertoFornecedor* e *MontaPedidoFornecedor*. As funções fazem a checagem e montagem dos pedidos do fornecedor, respectivamente. Por serem funções extensas, não faremos a implementação delas aqui. Elas poderão ser vistas baixando o código-fonte do site da *DevMedia* na sessão *ClubeDelphi*. Por fim chamamos *VerificaPedidoProduto* e *AdicionaPedidoProduto*, que já foram explicadas anteriormente.

Listagem 9. Montagem dos Pedidos

```
01 function Tfuncoes.MontaPedidos(): string;
02 var
03   sql: TSQLDataSet;
04   DM : TDMPedidos;
05   CodigoPedido: String;
06 begin
07   result := '';
08   CodigoPedido := '0';
09   try
10     sql := TSQLDataSet.Create(nil);
11     DM := TDMPedidos.Create(nil);
12     try
13       with sql do
14       begin
15         SQLConnection := DM.sqlConexao;
16         CommandText :=
17           'SELECT PRO_CODIGO, FOR_CODIGO, PRO_QNTDECOMPRAAUTO
               FROM PRODUTOS '+
18           'WHERE PRO_ESTOQUEATUAL <= PRO_ESTOQUEMINIMO ';
19         Open;
20         if not IsEmpty then
21           while not EOF do
22           begin
23             CodigoPedido :=
24               VerificaPedidoAbertoFornecedor(
25                 FieldByName('FOR_CODIGO').AsString);
26             if (CodigoPedido = '0') then
27             begin
28               MontaPedidoFornecedor(FieldByName('FOR_
                   CODIGO').AsString);
29               CodigoPedido :=
30                 VerificaPedidoAbertoFornecedor(
31                   FieldByName('FOR_CODIGO').AsString);
32             end;
33             if not VerificaPedidoProduto(FieldByName('FOR_
                CODIGO').AsString,
34               FieldByName('PRO_CODIGO').AsString) then
35             AdicionaProdutoPedido(CodigoPedido,
36               FieldByName('PRO_CODIGO').AsString,
37               FieldByName('PRO_QNTDECOMPRAAUTO').AsString);
38             Next;
39           end;
40         Result := 'Pedidos encaminhados aos fornecedores com
              sucesso!';
41       end;
42     finally
43       sql.free;
44       DM.free;
45     end;
46   except
47     Result := 'Problemas na montagem dos pedidos!';
48   end;
49 end;
```

Todas nossas regras da camada servidora estão na *unit* recém codificada, onde as funções e procedimentos presentes serão responsáveis pelo gerenciamento das informações, ou seja, todos os métodos estarão presentes nesta camada, caracterizando assim nossa camada de negócios. Todos os procedimentos e funções foram escritos manualmente, sem utilização de componentes, exceto a ligação com a base de dados onde herdamos o *SQLConnection* configurado anteriormente no *SOAP Server Data Module (DMPedidos)*.

Basicamente, todas as *functions* e *procedures* seguem o mesmo padrão de programação, como a criação de um componente do tipo *TSQLDataSet* em *runtime*, responsável pela execução dos comandos *SQL* retornando resultados de buscas ou ações realizadas. Após a execução das instruções, um *result* é atribuído às *functions*, e em seguida, os objetos criados em memória são liberados utilizando o *free*.

A partir destas funções, terminamos a idéia de desenvolvimento da aplicação *Server*. Ao compilarmos a o projeto será gerada uma DLL que poderá ser disponibilizada e hospedada em um servidor Web para que possa ser acessada através do protocolo HTTP. Hospedado o arquivo já poderemos iniciar o processo de codificação das aplicações da camada cliente.

Conclusão

Certamente o desenvolvimento em camadas tem crescido nos últimos tempos por ser mais escalável e de fácil implementação e manutenção. Como vimos, não há segredos ou complexidade no desenvolvimento da camada servidora. Criamos o banco de dados, desenvolvemos as funções que serão acessada pelas camadas cliente e geramos a DLL para distribuição.

Na próxima e última fase desse mini-curso, veremos como montar a parte cliente e fazer a distribuição da aplicação. Baseado nesta estrutura, você poderá aplicar outros conceitos úteis no gerenciamento de pedidos. Poderá ainda trabalhar na segurança necessária entre as camadas através do protocolo SOAP.

No próximo artigo faremos a conclusão do artigo criando os dois aplicativos que utilizaremos na camada cliente, deixando o sistema funcional. Abraço e até a próxima. ●

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Chegou o Sistema de Créditos DevMedia

Agora o **conteúdo completo** do nosso site está **ao seu alcance!**

A partir de agora todas as **2000 vídeo-aulas** do site DevMedia podem ser compradas individualmente.

Economia - Você não precisa mais assinar oito produtos diferentes para ter acesso ao conteúdo completo do site!

Todas as vídeos que você sempre quis ver a partir **R$ 0,75!**

Saiba mais sobre o Sistema de Créditos!

DevMedia GROUP

www.devmedia.com.br/creditos

Nesta seção você encontra artigos intermediários sobre Delphi Win 32 e Delphi for .NET

Controle Wizard no .net 2.0

Criando assistentes passo a passo com o novo controle Wizard

Nesse artigo veremos

- Controle Wizard;
- Criando Wizard em aplicações ASP.NET;
- Configurando WizardSteps.

Qual a finalidade

Conhecer o controle Wizard para criação de assistentes passo a passo em Web Sites;

Quais situações utilizam esses recursos?

Com o controle Wizard é possível criar assistentes passo a passo para as mais diversas tarefas on-line, como cadastros, por exemplo.

É muito comum encontrarmos interfaces de *wizard* em aplicações *Web*, ou seja, aplicações que orientam o usuário a realizar determinadas tarefas passo a passo. Os famosos *Next, Next, Finish*. Assim como na instalação de um software, temos vários passos a seguir, com a possibilidade de avançar ou retornar ao passo anterior.

Tradicionalmente, poderíamos criar uma interface *wizard* utilizando diversas páginas, onde cada uma é um passo do processo. Mas eis que na versão 2.0 do *ASP.NET* temos um controle específico para desenvolvimentos de interfaces *wizard*, facilitando em muito o trabalho de criar esse tipo de solução. Neste artigo veremos como utilizar e configurar o controle *Wizard* do *ASP.NET 2.0*.

Adriano Santos
falecom@adrianosantos.pro.br
É desenvolvedor Delphi desde 1998, professor e programador PHP, bacharel em Comunicação Social pela Universidade Cruzeiro do Sul, SP. É Editor Chefe das revistas ClubeDelphi e WebMobile. Gerente de TI na SoftPark, parceira Borland, membro fundador do DUG-SP (www.dug-sp.com) e mantenedor do blog Delphi to Delphi (www.delphitodelphi.blogspot.com) com dicas, informações e tudo sobre desenvolvimento Delphi.

Resumo do DevMan

Todos nós conhecemos o famoso Next, Next, Finish em instaladores de programas e até mesmo de sistemas operacionais. Mas como fazer isso na internet e por quê? O uso mais comum de assistentes na internet acontece em cadastros e alterações de preferências de sistema, como é o caso do Orkut.

Nesse artigo, faremos uso do controles Wizard do .net 2.0 com o RAD Studio 2007. Veremos como utilizá-lo na prática para criar uma pequena tela de cadastro de usuário.

Incluindo o controle Wizard na página

Vamos direto a um exemplo prático. Abra o seu *RAD Studio 2007* e crie um site usando o menu *File>New>ASP.NET Web Application – Delphi for .NET* e salve o projeto com o nome *Wizard*s. Veja que na *Tool Palette* em *Web Controls*, temos um controle chamado *Wizard* (**Figura 1**).

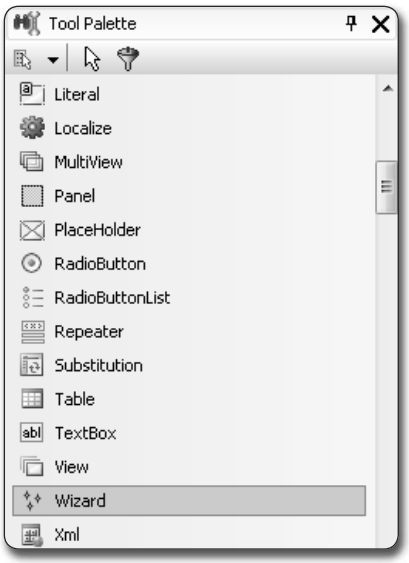


Figura 1. Controle Wizard na Tool Palette

Arraste um controle deste tipo para a página *Default.aspx* do projeto que acabamos de criar. Veja que ao incluir o controle na página, temos acesso ao *Wizard Tasks* (**Figura 2**).

Figura 2. Formatando layout do controle Wizard

Nele, a primeira opção é o *Auto Format*. Clique no *Auto Format* e troque a formatação para *Professional*.

Definindo os passos do Wizard

Um *wizard* pode ter inúmeras aplicações, o preenchimento de informações cadastrais para efetuar uma compra, o processo de configuração de um sistema, etc. Neste exemplo vamos criar um *wizard* que irá capturar as informações necessárias para cadastro em um site de relacionamento (tipo o "Orkut").

A primeira configuração que devemos fazer em um controle *Wizard*, é definir quais serão os passos (*steps*) da tarefa a ser executada. Para isso, nas *Tasks* do controle, clique na opção *Add/Remove WizardSteps* e será aberto um editor.

Esse editor é responsável pela manutenção dos *WizardSteps*, que representa um passo do *wizard*. Utilizando o botão *Remove*, exclua todos os *WizardSteps* que aparecem. Clique no botão *Add* para incluir o nosso primeiro *WizardStep*. Ele se chamará "Bem vindo!", que será uma introdução de boas-vindas do *wizard*.

Para configurarmos o nome do *WizardStep* é só modificar a propriedade *Title*, que fica à direita. Inclua mais três *Steps*, que se chamarão respectivamente: *Dados Pessoais*, *Dados Profissionais* e *Conclusão* (**Figura 3**). Feche o editor para voltar ao controle *Wizard*.

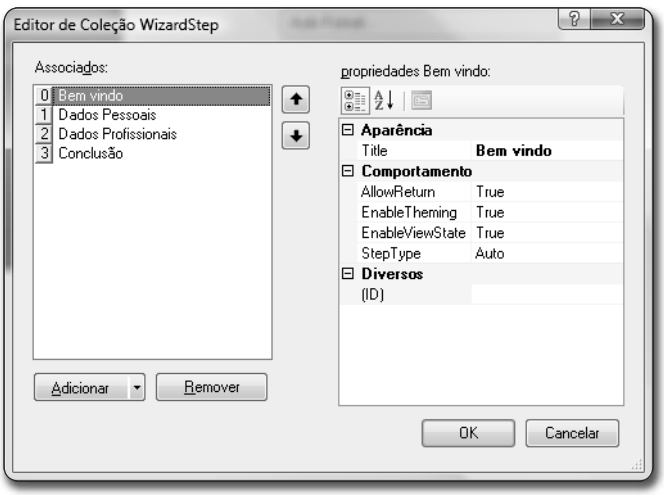


Figura 3. Configurando os passos do Wiizard

Nota do DevMan

Você deve ter observado que cada Step possui um ou mais botões. Através das Tasks do Wizard, utilizando a opção Step, você pode modificar qual WizardStep está aparecendo em nosso design. Veja que o primeiro Step possui o botão Next, o último possui os botões Finish e Previous. Os demais Steps possuem os botões Next e Previous. Temos a possibilidade de modificar os textos desses botões para trocarmos pelas respectivas palavras em português: Próximo, Anterior e Concluir. Para isso, vá nas propriedades do Wizard e modifique o valor das propriedades listadas a seguir:

- FinishCompleteButtonText: Concluir;
- FinishPreviousButtonText: Anterior;
- StartNextButtonText: Próximo;
- StepPreviousButtonText: Anterior;

Além de trocar o Text dos botões, através das propriedades, podemos realizar diversas customizações, incluindo configurações de estilos, formatações de layout, imagens para os botões, etc.

Incluindo texto no Bem Vindo

Cada *Step* de um *Wizard* funciona como um *Container*, onde podemos incluir controles e desenharmos assim cada passo do processo. No primeiro *Step* vamos apenas incluir um texto de boas- vindas.

Para isso, selecione o *Bem Vindo* através do *Wizard Tasks* clique na área vazia logo acima do botão *Próximo*. Inclua e formate um texto de Boas-Vindas nesse espaço. Veja na **Figura 4** como o primeiro *Step* do controle *Wizard* ficará.

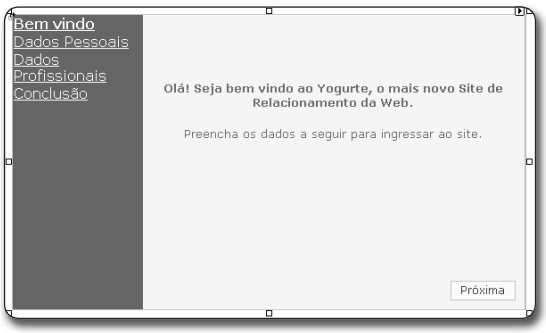


Figura 4. Incluindo texto no StepWizard

Criando um formulário no Wizard

Vamos agora criar um formulário para inclusão dos dados pessoais do usuário. Selecione o passo *Dados Pessoais* e clique na área vazia do *Step*. O cursor irá aparecer dentro do *Step*, vá então a barra de ferramentas superior do Delphi e clique no botão *Insert Table*. Inclua uma tabela com 2 colunas e 5 linhas. Veja na **Figura 5** com deve ficar o formulário para preenchimento das informações pessoais do usuário.

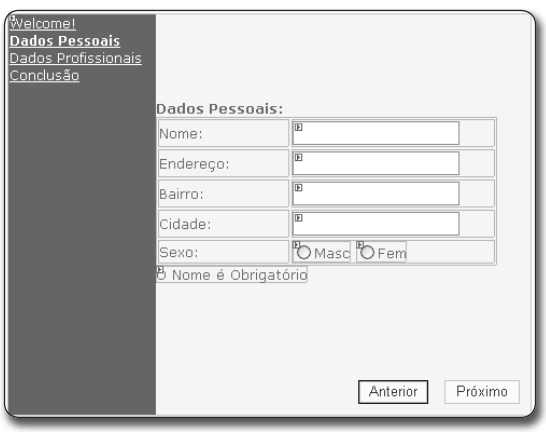


Figura 5. Criando o formulário de dados pessoais

Veja que utilizamos vários *TextBoxes* e dois *RadioButtons* para o campo *Sexo*. O importante aqui é que qualquer controle pode ser utilizado dentro de um controle *Wizard*. Devemos encarar cada *Step* como uma página comum, onde desenvolvemos nossos formulários.

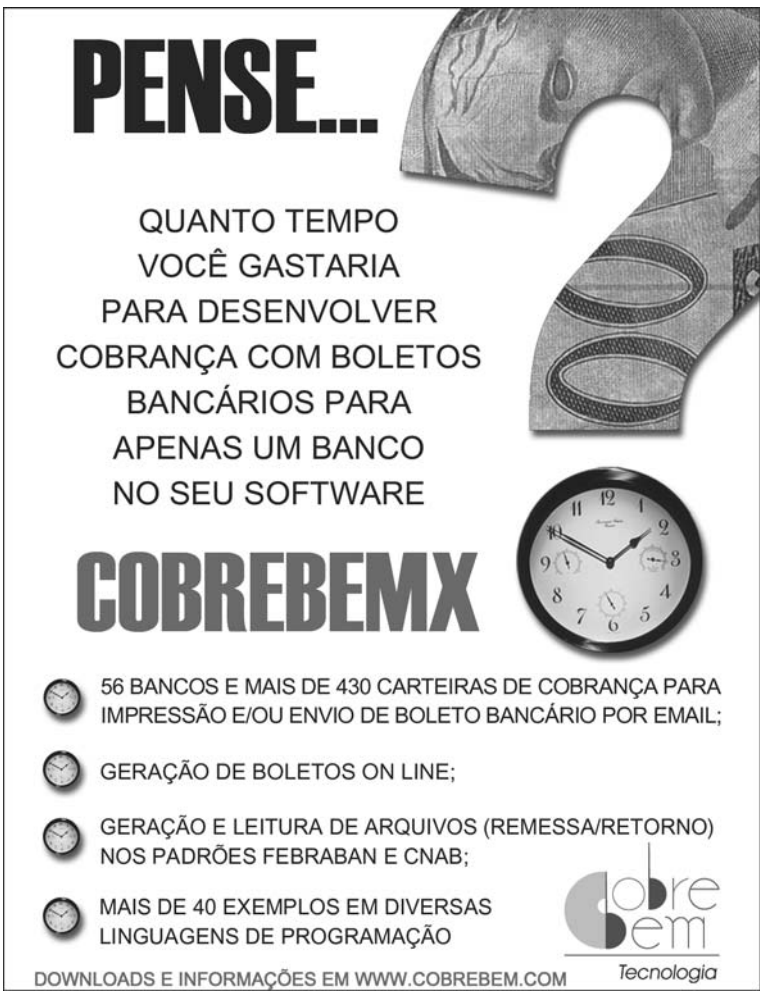
Também foi incluído um controle *RequiredFieldValidator* para validar o preenchimento do nome do usuário. Nas propriedades do *RequiredFieldValidator*, em *ControlToValidate* informamos o ID do *TextBox* referente ao nome do usuário. E na propriedade *ErrorMessage* incluímos o texto: "O Nome é obrigatório".

Para o passo *Dados Profissionais* vamos fazer a mesma coisa, crie uma tabela e inclua os controles necessários para construir um formulário, semelhante ao mostrado na **Figura 6**.

Figura 6. Criando um formulário de dados profissionais

Finalizando o Wizard

Em nosso *Step* de conclusão, vamos apenas incluir um texto de finalização. O importante aqui é que ação tomar quando o usuário clicar em *Concluir*. Nas propriedades do controle *Wizard*, acesse os



eventos do mesmo. Veja que podemos codificar diversos eventos possíveis, um deles é o *FinishButtonClick*. Dê um duplo clique no *FinishButtonClick* e veja que seremos redirecionados ao evento, pronto para ser codificado.

Aqui podemos tomar a ação que desejarmos com os dados. Se a nossa aplicação fosse real, o correto seria persistir os dados preenchid os em um banco de dados. Para efeito de testes vamos apenas mostrar uma mensagem com o nome do usuário. Veja na **Listagem 1** a codificação necessária para dispararmos uma mensagem de finalização ao usuário. O que deve ficar claro, é que nesse evento, podemos dar o destino que quisermos aos dados preenchidos.

Listagem 1. Mensagem de conclusão do Wizard

```
01 procedure TDefault.Wizard1_FinishButtonClick1(sender: System.
    Object; e:
02  System.Web.UI.WebControls.WizardNavigationEventArgs);
03 var
04   Texto : string;
05 begin
06   Texto := 'Olá ' + TextBox(Wizard1.FindControl('txtNome')).Text +
07     ' seu cadastro foi efetuado com sucesso!';
08   Response.Write(
09     '<script language=''javascript''>window.alert(''' +
10     Texto + ''')' + ';</script>');
11 end;
```

Testando a Aplicação

Deixe o *Welcome* selecionado para que seja o primeiro *Step* a aparecer para o usuário. Salve, compile e execute o projeto. Na tela *Welcome* clique no botão *Próximo*. Veja que o *Wizard* passa para o *Step* seguinte. Preencha as informações da tela *Dados Pessoais*.

Tente passar para o próximo *Step* sem preencher o campo Nome. Veja que o controle *Wizard* só permite que você passe ao próximo passo se o controle de validação não acusar nenhuma inconsistência. Veja também que é possível navegar pelo *Wizard* através dos links que temos na barra lateral esquerda e não apenas pelos botões.

Dica: Para ocultar os links no painel esquerdo e permitir a navegação somente pelos botões, configure para *False* a propriedade *DisplaySideBar*.

Clique em *Próximo*, preencha os *Dados Profissionais* e clique novamente no botão *Próximo*. No final, clique no botão *Concluir* e veja que a mensagem de conclusão do controle *Wizard* será mostrada. Veja um exemplo na **Figura 7**, lembrando que aqui poderíamos persistir os dados informados em um banco de dados e redirecionar o usuário para uma outra página do *site*.

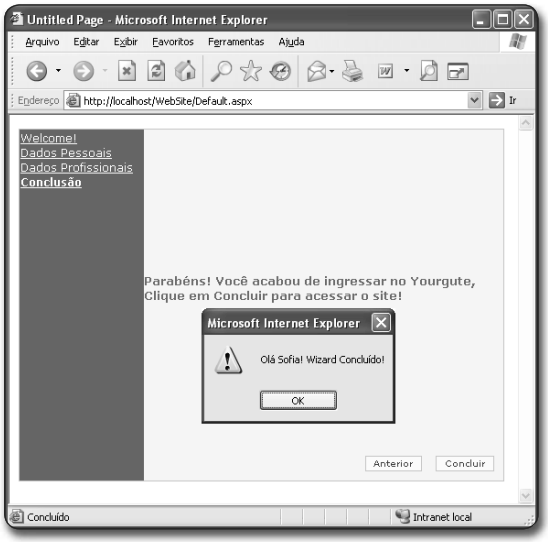


Figura 7. Concluindo Wizard

Nota do DevMan

Um WorkFlow, ou simplesmente fluxo de trabalho é, segundo o Wikipédia, a seqüência de passos necessários para que se possa atingir a automação de processos de um negócio, de acordo com um conjunto de regras definidas, envolvendo a noção de processos, permitindo que estes possam ser transmitidos de uma pessoa para outra de acordo com algumas regras. Um exemplo de Workflow está no processo de compra online. Desde o ponto onde você confirma sua compra até você receber o produto, existe uma série de passos que devem ser realizados, como solicitação do produto ao setor de estoque e este na seqüência envia o produto para o setor de empacotamento, que por sua vez envia ao setor de distribuição onde será então entregue à uma transportadora e finalmente você recebe o produto.

Conclusão

Novamente pudemos conferir como o *ASP.NET 2.0* facilita o trabalho do desenvolvedor. Criamos uma estrutura de *wizard* com praticamente nenhum código. O único código utilizado aqui foi para dispararmos uma mensagem ao final.

Podemos criar soluções mais avançadas de *wizard*, como por exemplo, modificar os *Setps* em tempo de execução. Imagine uma solução de *Workflow*, onde cada etapa do processo dependa de informações inseridas pelo usuário. Explorando as propriedades e eventos do controle, podemos aplicar o conceito de *wizard* da maneira mais apropriada aos nossos requisitos. ●

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Nesta seção você encontra artigos sobre técnicas que poderão aumentar a qualidade do desenvolvimento de software

Padrões de Projeto – Parte 1

Entendendo os principais padrões de desenvolvimento de sistemas

Nesse artigo veremos

- O que são padrões de projeto e como estão organizados;
- Como implementá-los em Delphi;
- Como utilizá-los em nosso dia-a-dia através de exemplos práticos.

Qual a finalidade

Mostrar o uso adequado de padrões de projeto em uma aplicação Delphi.

Quais situações utilizam esse recurso?

O desenvolvimento de aplicações utilizando-se de padrões pode ser feito em qualquer tipo de sistema, desde aplicações Win32 até complexos softwares de gerenciamento.

Paulo Quicoli

pauloquicoli@devmedia.com.br

Editor técnico da revista ClubeDelphi, formado em processamento de dados pela FATEC-TQ. Atua como analista e desenvolvedor da ControlM Informática (www.controlm.com.br). Utiliza Delphi desde de sua primeira versão, entusiasta do desenvolvimento orientado a objetos tem publicado vários artigos sobre o assunto. Blog: www.pauloquicoli.spaces.live.com .

Os padrões de projeto, ou *design patterns*, como são também conhecidos, estabelecem um conjunto de soluções para problemas que sempre aparecem ao se desenvolver um sistema, especialmente utilizando orientação a objetos. Esses problemas foram agrupados em três categorias: *criacional, estrutural* e *comportamental*. Cada categoria contém uma série de padrões envolvidos, vou listá-los aqui a caráter de informação.

- Padrões criacionais
  - Abstract Factory;
  - Buil der;
  - Factory Method;
  - Prototype;
  - Singleton.

Resumo do DevMan

Os padrões de projeto são soluções para problemas recorrentes que surgem no desenvolvimento de sistemas. Foram inicialmente catalogados pela GoF (Gang of Four). Quatro autores, Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides, que lançaram o livro mais famoso sobre o assunto: Design Patterns: Elements of Reusable Obejct-Oriented Software. Padrões de projeto são considerados como boas práticas entre os desenvolvedores, por isso veremos como aplicar alguns desses padrões em Delphi, em situações reais que nos afligem no dia-a-dia.

- Padrões estruturais
  - Adapter;
  - Bridge;
  - Composite;
  - Decorator;
  - Façade;
  - Flyweight;
  - Proxy.

- Padrões comportamentais
  - Chain of Responsability;
  - Command;
  - Interpreter;
  - Iterator;
  - Mediator;
  - Memento;
  - Observer;
  - State;
  - Strategy;
  - Template Method;
  - Visitor.

Nesta série de artigos quero abordar cada grupo de padrões e criar um exemplo prático que demonstre utilização de alguns deles.

Padrões criacionais

Nesta categoria de padrões encontramos soluções no que diz respeito à criação de objetos. Por exemplo, em como manter uma única instância de um objeto. Um dos padrões de desenvolvimento mais conhecido e que é bastante abordado no dia-a-dia é o *Singleton*. Vejamos como implementar soluções utilizando esse padrão de forma bastante prática.

Digamos que um determinado tipo de objeto de sua aplicação só pode ser instanciado uma única vez e então ser utilizado por toda a aplicação. Como resolvemos isso? Vamos aplicar o padrão *singleton* para garantir que esse objeto só exista uma única vez e que também tenhamos um ponto de acesso único a ele, ou seja, vamos acessar esse objeto apenas pelo *singleton*.

Inicie um novo projeto Win32 no *Delphi 7,* ou qualquer outra versão que preferir, usando o menu *File>New>Application* e salve-o como *PadraoSingleton*. Observe que um *Project Group* é criado por padrão. Vamos utilizá-lo para manter posteriormente os outros exemplos que virão. Desta forma teremos um catálogo para consulta. Vamos salvar o *Project Group* como *Padrões*. Para isso clique com o botão direito do mouse sobre ele escolha *Save Project Group*.

Nota

É importante lembrar que os padrões aqui demonstrados não estão associados ou ligados a determinada versão do Delphi, ou seja, esse artigo foi criado com o auxílio do Delphi 7, mas todos os conceitos e estruturas de código podem ser totalmente utilizados em qualquer outra versão da ferramenta.

Nosso *singleton* irá manter a instância de uma classe responsável pela leitura e gravação no registro do *Windows*. Adicione uma nova *unit* ao projeto utilizando o menu *File>New>Unit* e salve-a como *SingletonU.pas*. Nela vamos adicionar as seguintes classes vistas na **Listagem 1**.

Listagem 1. Estrutura das classes

```
IRegistro = interface
  ['{E89DC81B-6004-4DB1-B5ED-3227B67A8915}']
  function LeRegistro(Nome: string): string;
  procedure GravaRegistro(Nome: string; Valor: string);
end;

TRegistro = class(TInterfacedObject, IRegistro)
  public
    function LeRegistro(Nome: string): string;
    procedure GravaRegistro(Nome: string; Valor: string);
end;

TRegistroSingleton = class
  public
    class function Instance : IRegistro;
end;

const
  Chave = 'Software\Padroes';

var
  RegistroGlobal: IRegistro;
```

Nota

A leitura e gravação do registro do Windows não será comentada aqui por não fazer parte do escopo do artigo.

A implementação do padrão é feita na classe *TRegistroSingleton*. Veja que ela possui um método chamado *Instance*. Esse método é responsável por instanciar um objeto que implemente a interface *IRegistro* e manter apenas uma única instância existente. No código a seguir podemos ver como isso é feito. Repare que fazemos uma checagem na variável *RegistroGlobal* para verificar se ela está instanciada. Caso não esteja, seu conteúdo será *nil* indicando que não há nenhuma instância da variável em memória.

```
class function TRegistroSingleton.Instance:
IRegistro;
begin
  if RegistroGlobal = nil then
    RegistroGlobal := TRegistro.Create;
  Result := RegistroGlobal;
end;
```

Nota do DevMan

As interfaces são estruturas de linguagens orientadas a objetos que definem uma espécie de contrato que deve ser obedecido por quem vai utilizá-las ou implementá-las. Nelas não construímos os métodos, apenas os definimos. Quem implementa uma determinada interface é outra classe que tem por obrigação implementar todos os métodos que são expostos. Vimos isso com clareza no código anterior onde temos na interface IRegistro a definição de dois métodos e em sua implementação, feita na classe TRegistro vemos ambos construídos.

ClubeDelphi PLUS www.devmedia.com.br/clubedelphi/portal.asp

Acesse agora o mesmo o portal do assinante ClubeDelphi e veja uma vídeo-aula de Renato Matos que mostra como trabalhar com interfaces.

http://www.devmedia.com.br/articles/viewcomp.asp?comp=10096

A variável *RegistroGlobal* é uma variável pública que é utilizada pelo padrão para que se retorne sempre uma única instância. Vamos ver como utilizamos agora nosso *singleton*.

Adicione ao formulário principal três botões, alterando sua propriedade *Caption* para *1. Ler valores padrão*, *2. Alterar valores padrão* e *3. Ler valores alterados*, respectivamente. Na **Listagem 2** vemos o *onClick* do botão 1.

Listagem 2. Utilizando o padrão

```
procedure TForm1.Button1Click(Sender: TObject);
var
 Registro: IRegistro;
begin
  Registro := TRegistroSingleton.Instance;
  ShowMessage(Registro.LeRegistro('Valor1'));
  ShowMessage(Registro.LeRegistro('Valor2'));
end;
```

Observe que não instanciamos diretamente um objeto que implemente *IRegistro*, solicitamos à classe *TRegistroSingleton* que nos retorne uma instância dele. Ao *debugarmos* o método *Instance* vemos que na primeira vez que ele é chamado a variável *RegistroGlobal* é igual à *nil* e então é criada, porém na segunda vez ela é diferente dessa condição, portando não é criada novamente, sendo apenas retornada. Isso é visto nas **Figuras 1** e **2**.

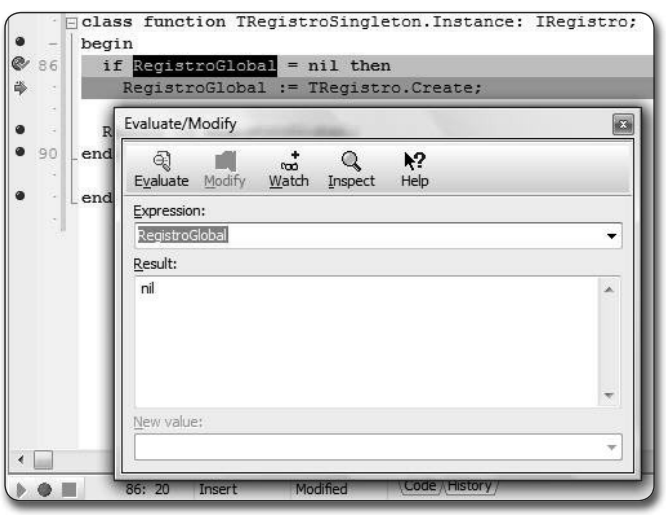


Figura 1. Variável RegistroGlobal é nil na primeira execução

Como pode ser visto, a implementação de um *singleton* em *Delphi* nada mais é de que gerenciar a criação de variáveis públicas. Você pode se perguntar agora porque não utilizar apenas a variável pública. Um dos fatores que levam ao uso do padrão *singleton* é economia de memória. Veja que sempre retornamos a mesma instância da variável pública, não a deixamos ser criada mais de uma vez dentro do código.

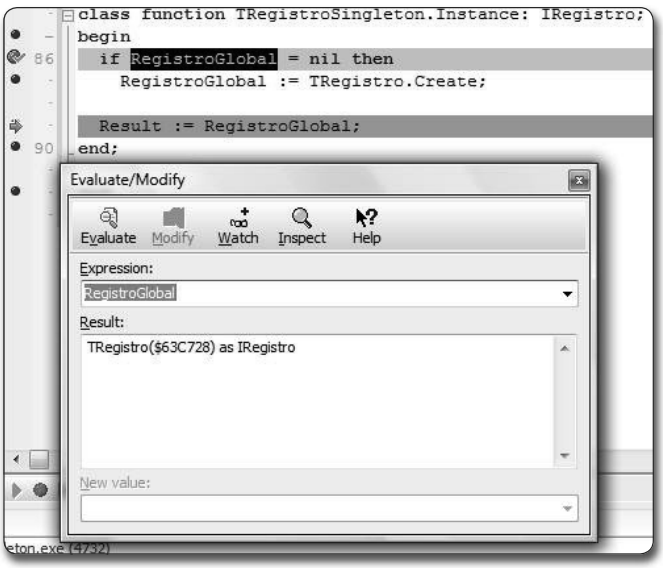


Figura 2. Variável RegistroGlobal não é mais nil nas execuções subseqüentes

Nota do DevMan

Todo objeto durante a execução de um aplicativo recebe um endereçamento de memória. Porém esse endereçamento só é atribuído no momento em que instanciamos esse objeto. Enquanto não for instanciado ele possui um endereçamento nulo que em cada linguagem possui uma constante diferente. Em Delphi esse valor nulo é representado pela constante nil, em C# por exemplo, é null em Visual Basic é nothing, e assim por diante. Portanto, quando um objeto é igual à nil significa que ele ainda não foi instanciado.

Padrão Builder

O *Builder* possui a tarefa de encapsular a construção complexa de um objeto. Por complexa podemos dizer por exemplo a necessidade de se relacionar vários objetos dependentes. Para que essa situação fique mais clara, vamos criar o segundo exemplo do nosso projeto. Adicione ao *Project Group* um novo projeto. Para isso clique com o botão direito no nome do *Project Group* e escolha *Add New Project>VCL Forms Application*. Uma nova aplicação *Win32* será criada. Vamos salvá-la com o nome de *Builder*. A esse projeto *Builder* vamos adicionar agora um *DataModule* e criar uma conexão *IBX* com o banco de exemplo que vem com o *Firebird*, o *Employee.fdb*. O que quero criar aqui é uma situação rotineira, onde criamos um objeto *TIBQuery* dinamicamente para realizar alguma forma de consulta. O porém é que sempre que criamos um *TIBQuery* precisamos associar a ele um objeto do tipo *TIBDatabase* e *TIBTransaction*. Como já foi mencionado, o objetivo do padrão *Builder* é esconder essa complexidade que está definida pela necessidade de se ligar dois objetos no nosso caso, do restante do sistema, ou seja, quem saberá como instanciar um *TIBQuery é o Builder*. Dessa forma centralizamos a criação do objeto em um único lugar,

assim, caso algo mude nessa criação vamos alterar apenas no *Builder* e não o sistema todo. Vamos a prática:

Com nosso *DataModule* já conectado ao banco de dados vamos adicionar uma nova *unit* ao projeto e salvá-la como *PadraoBuilder.pas*. Adicione a ela o código da **Listagem 3.**

Listagem 3. Builder de TIBQuery

```
unit PadraoBuilder;

interface

uses DataModule1U, IBQuery;

type
  TIBQueryBuilder = class
    class function CreateInstance: TIBQuery;
  end;

implementation

{ TIBQueryBuilder }

class function TIBQueryBuilder.CreateInstance: TIBQuery;
var
  lquery: TIBQuery;
begin
   lquery := TIBQuery.Create(nil);
   lquery.Database := DataModule1.IBDatabase1;
   lquery.Transaction := DataModule1.IBTransaction1;
   result := lquery;
end;

end.
```

Observe que na seção *uses* adicionamos uma referência ao nosso *DataModule* e também à *unit IBQuery*, que é onde o tipo *TIBQuery* está definido. Posteriormente criamos a classe *TIBQueryBuilder* e adicionamos a ela um método *CreateInstance*. Veja agora o código desse método, ele apenas cria um objeto do tipo *TIBQuery*, faz a associação necessária entre ele e os objetos *IBDatabase1* e *IBTransaction1* que estão no *DataModule1* para que finalmente seja retornado. Essa é a estrutura de um *Builder*. Para vermos em ação vamos adicionar um botão ao formulário do projeto e nele vamos criar uma consulta dinâmica que retorne a quantidade de registros da tabela *Employee* do banco de dados. Na **Listagem 4** vemos como isso é feito.

Observe que nesse momento não adicionamos referência ao *DataModule*, visto que quem o utiliza é o nosso *Builder* que está implementado na *unit PadraoBuilder*. O momento exato em que o padrão é utilizado é quando instanciamos a variável *lquery*.

```
Listagem 4. Builder em execução
uses PadraoBuilder, IBQuery;

{$R *.dfm}

procedure TBuilderF.Button1Click(Sender: TObject);
var
  lquery: TIBQuery;
begin
  lquery := TIBQueryBuilder.CreateInstance;
  lquery.SQL.Add('SELECT COUNT(*) AS TOTAL FROM EMPLOYEE');
  lquery.Open;
  ShowMessage('A quantidade registros na tabela EMPLOYEE é de: '+
              lquery.FieldByName('TOTAL').AsString);
  lquery.Close;
  FreeAndNil(lquery);
end;
```

```
lquery := TIBQueryBuilder.CreateInstance;
```

Nesse momento não temos mais que nos preocupar com o que é necessário para que um *TIBQuery* funcione, isso agora é responsabilidade da classe *TIBQueryBuilder*.

Conclusão

O uso dos padrões de projeto acaba por organizar melhor nosso código porque aplicam um conceito muito almejado no desenvolvimento orientado a objetos, o *SoC (Separation of Concerns)*. Uma tradução livre para ele é separação de responsabilidades. Cada procedimento ou função que compõem uma rotina tem sua responsabilidade bem definida e só faz o que tem que fazer. Um exemplo disso é visto no exemplo do *Builder*. Veja que ao aplicá-lo retiramos da *procedure Button1Click* a necessidade dela ter que ligar o *IBDatabase1* e o *IBTransaction1* ao objeto *lquery*.

Nos próximos artigos continuaremos a explorar as outras categorias de padrões e como aplicá-los, mesmo que nossos sistemas não sejam totalmente orientados a objeto. ●

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Nesta seção você encontra artigos sobre técnicas que poderão aumentar a qualidade do desenvolvimento de software

Desenvolvimento em Equipe

Utilizando Subversion para versionamento de código fonte

Nesse artigo veremos
O que é o Subversion;
Como instalá-lo;
Como utilizá-lo em nosso dia a dia.

Qual a finalidade
Mostrar o uso adequado de versionamento de código fonte, seja em uma equipe ou mesmo individualmente.

Quais situações utilizam esse recurso?
No desenvolvimento de sistemas em geral.

Porque preciso controlar as versões do meu código fonte? Durante o desenvolvimento de um sistema alteramos várias vezes os códigos-fonte. Mesmo depois finalizado, durante a fase de manutenção, continuamos as alterações e não ter controle sobre isso pode ser um grande problema, mesmo trabalhando individualmente. Vou ilustrar aqui algumas situações que podem acontecer, ou melhor, que já aconteceram comigo mesmo há um bom tempo atrás.

Imagine que você tenha realizado uma série de ajustes no sistema, todos solicitados pelo seu cliente. Logo depois da entrega dos recursos prontos esse cliente resolve que não deseja mais as alterações e que tudo volte como era antes. O que você faz? Vol ta um *backup*? Vamos para uma situação um pouco mais complicada. Você está implementando uma série de recursos novos em seu sistema,

Paulo Quicoli
pauloquicoli@devmedia.com.br
Editor técnico da revista ClubeDelphi, formado em processamento de dados pela FATEC-TQ. Atua como analista e desenvolvedor da ControlM Informática (www.controlm.com.br). Utiliza Delphi desde de sua primeira versão, entusiasta do desenvolvimento orientado a objetos, tem publicado vários artigos sobre o assunto. Blog: www.pauloquicoli.spaces.live.com .

Resumo do DevMan

Controlar as versões de um sistema hoje é fundamental, ainda mais em equipe. Neste artigo vemos como o Subversion, ou simplesmente SVN, pode nos ajudar. Entendendo sua estrutura de repositórios, que criam um ponto comum de compartilhamento de informações, podemos aplicar estratégias para sempre ter uma versão do código fonte pronta e estável, independente da quantidade de requisitos ou de programadores envolvidos no projeto.

porém ainda não estão finalizados. Seu cliente então entra em contato com você e precisa que urgentemente seja feita uma alteração devido a alguma mudança na regra de negócio. Só que você tem uma série de códigos não finalizados e não testados que impedem a imediata solução do problema do seu cliente. O que você faz nesse caso? E por último, agora você tem um amigo lhe ajudando no desenvolvimento. Como que você gerencia quando ele precisa alterar uma *unit* que você está utilizando no momento? Você pede para ele esperar?

Essas situações podem parecer excepcionais, mas são mais comuns do que imaginamos, principalmente para quem está começando e não possui muita experiência. Tudo isso pode ser resolvido utilizando um sistema controlador de versões como o *Subversion*.

Nesse artigo veremos como fazer uso o *Subversion* e melhorar o acompanhamento e gerenciamento dos fontes de nossas aplicações.

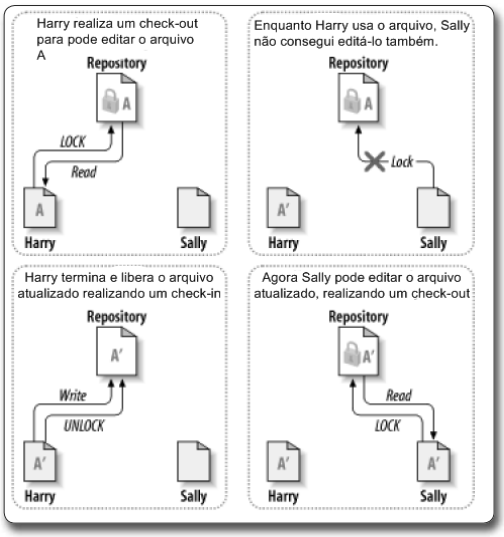


Figura 1. A solução *Lock-Modify-Unlock*

Conhecendo o Subversion

O *Subversion* é um sistema gerenciador de versões, não só de código fonte, mas de qualquer tipo de arquivo. Ele é *free* e *open-source* e trabalha no formato *cliente/servidor*, sendo ele a figura de servidor centralizando e compartilhando informações. As informações armazenadas nele são organizadas em *repositórios* que por sua vez são acessados pelos clientes. Mas como o *Subversion* pode nos ajudar a resolver as situações anteriormente mencionadas?

Basicamente duas estratégias são empregadas, a *Lock-Modify-Unlock* e a *Copy-Modify-Merge*. Vamos entender cada uma.

Lock-Modify-Unlock

Nessa solução quando precisamos alterar um arquivo, o recuperamos do repositório (*check-out*) e mudamos seu *status* para *travado*. Se um segundo usuário tentar recuperar do repositório o mesmo arquivo que estamos utilizando, uma mensagem de erro é reportada dizendo que o arquivo está travado. Após termos feito as alterações precisamos devolver o arquivo atualizado ao repositório (*check-in*), dessa forma o *status* do arquivo passa a ser *destravado* e outros usuários poderão utilizá-lo. Todo esse processo pode ser visto na **Figura 1**.

Nota

Sempre que uma nova versão de arquivo é enviada ao repositório através de uma operação de check-in, a anterior não é eliminada. Ela fica armazenada permitindo ao usuário restaurá-la a qualquer momento, através de uma operação de check-out.

Esta solução parece ser a ideal já que enquanto você está alterando um determinado arquivo, o mesmo está travado exclusivamente para você. Mas por traz dessa segurança existe a falha humana. Imagine que o arquivo que você precisa alterar foi travado pelo seu colega ao lado e ele saiu para o almoço. E agora? Ou numa situação mais real, você está finalizando uma série de alterações e várias *units* estão travadas para você e seu colega ao lado precisa corrigir um problema urgente exatamente em alguma *unit* utilizada por você. Veja que enquanto alguns problemas são solucionados outros surgem. Vamos ver a próxima solução.

Copy-Modify-Merge

Outros sistemas de controle de versões como *StarTeam* e *CVS* utilizam o modelo *copy-modify-merge* como alternativa ao sistema de *lock*. Esse modelo trabalha da seguinte forma: cada cliente solicita ao servidor uma cópia local do repositório, o que é conhecido como *working copy*. Então todos trabalham simultaneamente e independentemente em suas cópias de trabalho para posteriormente essas cópias sejam unidas em uma nova versão final. Na **Figura 2** vemos a primeira parte desse processo.

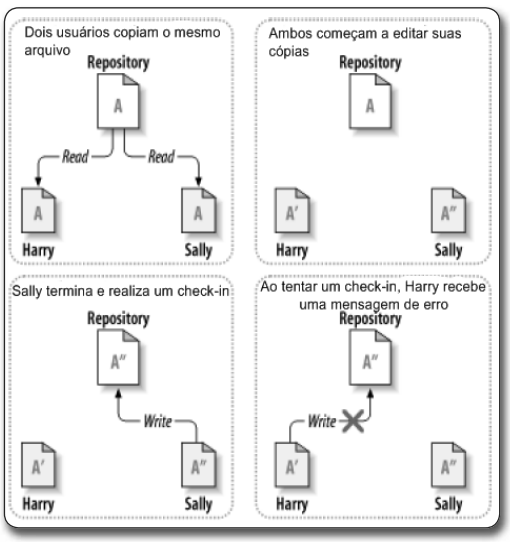


Figura 2. Processo Copy-Modify-Merge parte 1

ClubeDelphi PLUS www.devmedia.com.br/clubedelphi/portal.asp

Acesse agora o mesmo o portal do assinante ClubeDelphi e assista a uma vídeo aula de Renato Matos que mostra como instalar o StarTeam.

http://www.devmedia.com.br/articles/viewcomp.asp?comp=8396

Observando a **Figura 2** vemos que ambos, Harry e Sally, puderam alterar o mesmo arquivo independentemente um do outro. Sally por sua vez encerrou seu trabalho primeiro e realizou uma operação de *check-in*, tornando sua versão do arquivo a mais recente e atual no repositório. Quando Harry termina seu trabalho e tenta realizar também o *check-in* de suas alterações, o sistema de controle de versões dispara um erro alertando Harry que uma nova versão do arquivo que ele está tentando adicionar já foi incluída no servidor e que será necessário realizar um merge dessas versões.

Uma operação de merge nada mais é do que mesclar as suas alterações na versão mais recente do arquivo. Esse tipo de operação é muito comum em um ambiente de trabalho compartilhado, quando temos mais de uma pessoa em um projeto. Contudo uma operação de merge pode necessitar de uma comunicação entre os desenvolvedores envolvidos, visto que códigos serão mesclados. Na **Figura 3** vemos a finalização do processo de *copy-modify-merge*.

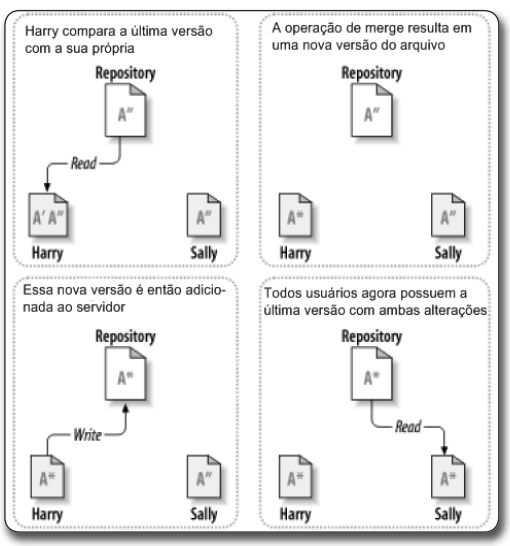


Figura 3. Processo Copy-Modify-Merge parte final

Nota

Sistemas de controle de versão, como o Subversion, disponibilizam ferramentas de apoio para a realização de mesclagem. Porém é preciso ter sempre em mente que mesmo que essas ferramentas sugiram uma forma de mesclar o código, essa forma deve ser avaliada pelo desenvolvedor porque somente ele é apto a resolver qualquer conflito existente entre as versões.

Com as duas soluções expostas, particularmente escolho a segunda porque interfere menos no dia a dia de uma equipe e até mesmo por fim acaba forçando com que os desenvolvedores se comuniquem mais quando um conflito de versão existir.

Instalando o Subversion

A versão disponibilizada pelo site *http://subversion.tigris.org/* possui uma instalação um pouco trabalhosa, requerendo até mesmo um servidor *Apache* para hospedar o *Subversion*, por isso decidi utilizar uma versão personalizada para o sistema operacional *Windows*, possuindo fácil instalação. Essa versão é *free* e encontra-se no site *www.visualsvn.com*. Após o devido *download* e execução do instalador, durante a instalação nos deparamos com a tela da **Figura 4**.

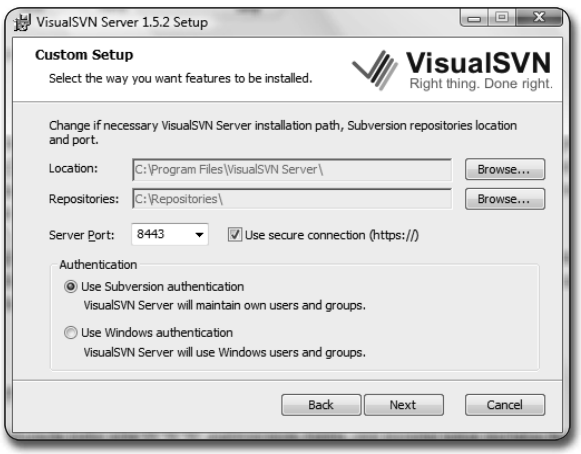


Figura 4. Configurando o servidor *Subversion*

Na tela da **Figura 4** podemos definir em que local o repositório de nossos projetos ficarão armazenados e também podemos já definir qual porta de comunicação será utilizada entre clientes e servidor. Sem esquecer que podemos também alterar a forma de autenticação desses clientes. Por enquanto, vamos deixar essas informações como estão e avançar a instalação. Nesse momento nenhuma configuração adicional será solicitada e ao finalizar teremos o gerenciador do *Subversion*, também conhecido como *SVN*, rodando como pode ser visto na **Figura 5**.

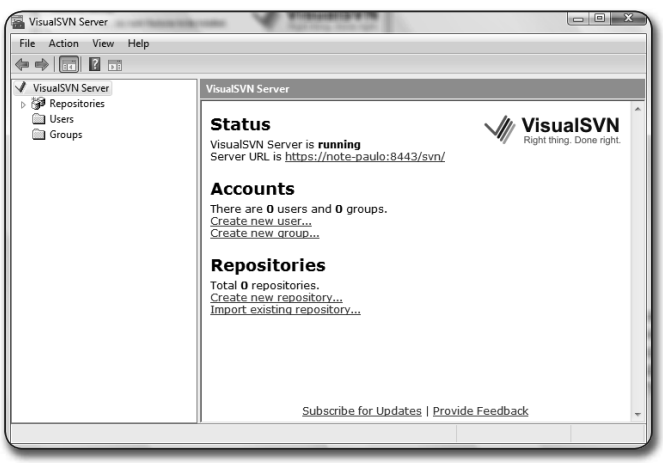


Figura 5. Gerenciado SVN em execução

Nota

O servidor SVN é instalado como um serviço do Windows, portando para utilizar o SVN não é preciso manter seu gerenciado aberto.

Como mencionado anteriormente, o *SVN* trabalha em arquitetura *Cliente/Servidor*. Para que esses clientes possam acessar o repositório é preciso criar contas de usuários do *SVN*. Como pode ser visto na **Figura 5** temos a informação de que existem 0 (zero) usuários e 0 (zero) grupos cadastrados. Vamos criar dois usuários, Paulo e Carla por exemplo.

Um repositório nada mais é que um diretório um pacote onde ficam armazenados os nossos arquivo. Precisamos ter um criado para que possamos encaixar os usuários e dar as devidas permissões. Portanto clique em *Create new repository* e lhe dê o nome de *ProjetoTeste*. Com o repositório criado, é possível definir quais usuários podem ter acesso e que tipo de acesso será. Por padrão, todos usuários do *SVN* possuem autorização para leitura e escrita, mas isso pode ser alterado acessando as propriedades do repositório desejado. Veja isso na **Figura 6**.

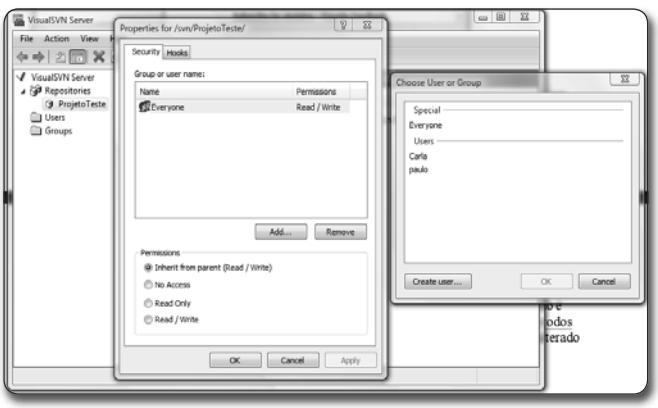


Figura 6. Implementando a segurança do repositório

Estruturando e utilizando o repositório

Ter o servidor *SVN* e um repositório criado não quer dizer que estamos prontos. Como enviamos alguma informação pra esse repositório? Para isso utilizamos os clientes do SVN. Um cliente neste caso é um software que se comunica com o servidor *SVN* e permite que nós o utilizemos. Existem vários clientes disponíveis, para os mais diversos ambientes e *IDEs*. Dentre todos um merece destaque, o *TortoiseSVN*. Para baixá-lo acesse *http://tortoisesvn.tigris.org/*.

Sua instalação é bem simples também e não exige configuração extra alguma. Ao término da instalação será necessário reiniciar o computador, isto porque o *TortoiseSVN* se integra ao *Explorer do Windows*. Agora precisamos estruturar nosso repositório previamente criado. Através do *Windows Explorer* acessamos o a pasta geral de repositórios, *C:\Repositories*, e nela encontramos *ProjetoTeste*. Clique com o botão direito do mouse sobre a pasta *ProjetoTeste* e no menu de contexto escolha *TortoiseSVN>Repo-Browser*.

O aplicativo que vai surgir é uma espécie de *gerenciador de repositórios*. Através dele vamos criar a estrutura mais comum quando se diz respeito ao uso do *SVN*. Vamos criar três pastas: *trunk*, *branches* e *tags*. Essa estrutura é quase um padrão entre os usuários do *SVN*. Cada uma dessas pastas possui uma função específica. Em *trunk* ficará todo o fonte base, funcional e estável do projeto. Em *branches* ficarão as cópias dos códigos-fonte que usaremos para desenvolver alguma funcionalidade. E por fim, em *Tags*, teremos cópias representando algum ponto no tempo, por exemplo alguma versão específica do projeto. Veja na **Figura 7** como ficou nossa estrutura.

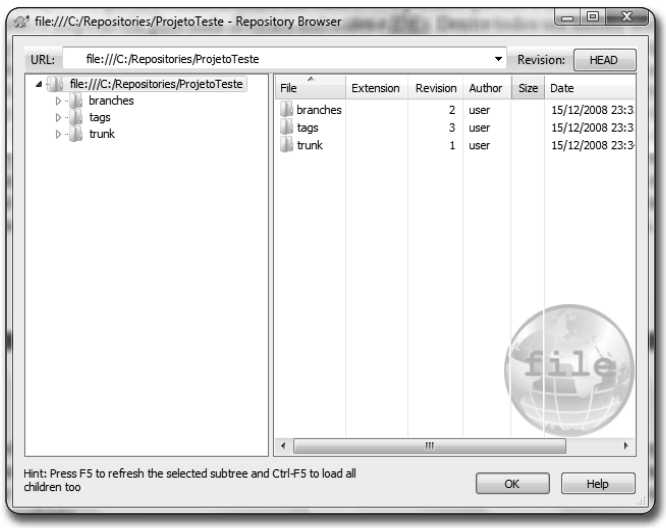


Figura 7. Estrutura do repositório ProjetoTeste

Chegou a hora de por tudo isso pra funcionar. Crie um novo projeto do tipo *VCL Forms Application*. Salve-o em uma pasta chamada *Teste*. Vamos adicionar o código-fonte inicial ao repositório, porém é preciso excluir alguns arquivos do versionamento, por exemplo os *.dcu*, *.local*, *_history*, *.exe* e outros arquivos temporários que são gerados.

Clique com o botão direito na pasta *Teste* e no menu de contexto escolha *TortoiseSVN>Settings*. Na janela que surgir vamos incluir no campo *Global Ignore Pattern* os arquivos que deverão ser ignorados. Podemos utilizar máscaras para isso, apenas tomando o cuidado de que entre uma máscara e outra haja apenas um espaço em branco para separá-las.

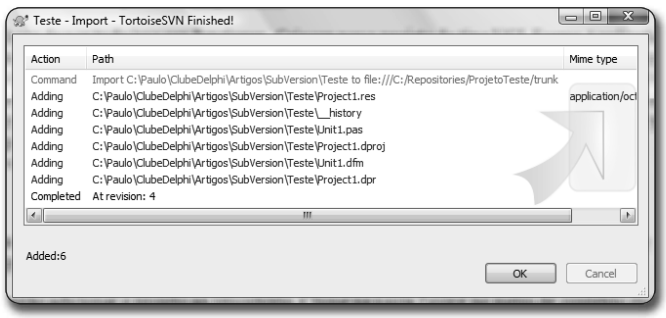


Figura 8. Inclusão bem sucedida no repositório

Nota

Uma sugestão de arquivos que podem ser ignorados no versionamento: *.dcu *.~dfm *.~pas *.exe *.local _history *.ddp *.~ddp

Podemos então adicionar o projeto ao repositório. Clique na pasta *Teste* e no menu de contexto escolha *TortoiseSVN>Import*. Na janela de importação clique no botão que contém reticências e escolha a pasta *trunk* do repositório. Ao término da inclusão do código-fonte, uma janela de confirmação é exibida, como visto na **Figura 8**.

Simulando o dia-a-dia

Para entender na prática como que a solução *copy-modify-merge* pode nos ajudar, vamos simular aqui o dia-a-dia de uma equipe de desenvolvimento. Quero já deixar claro que esta é uma simulação, portanto nem todos os aspectos reais poderão ser abordados, mas os mais comuns serão vistos.

Implementando funcionalidades

O cliente do projeto *Teste* precisa que no formulário principal exista um botão que ao clicar nele seja exibida a data atual. A pessoa responsável por essa implementação será a Carla. No momento ela ainda não tem o código-fonte do projeto em sua máquina. Ela precisa recuperar a versão mais atual deles. Então ela cria uma pasta para o projeto, por exemplo em *C:\Carla* e cria a pasta *Projeto*. Ao clicar com o botão em *Projeto*, no menu de contexto é escolhido *SVN Checkout*. Na tela de *check-out* é possível escolher qual repositório e pasta que contém a versão que desejamos recuperar e também podemos informar para onde copiar esses arquivos. Observe que vamos recuperar os arquivos do *trunk*, pois é nele que estão os fontes finais e estáveis.

Nota

Veja que não há necessidade de que todos os desenvolvedores tenham o projeto em uma mesma pasta, definida em um mesmo local. Ex.: O projeto Teste deve ser gravado sempre em: D:\Projetos\Teste. Isso não é obrigatório para o SVN.

Após o *check-out* é possível ver pelo *Windows Explorer* a situação dos arquivos. Veja na **Figura 9** a integração do *TortoiseSVN* com o *Windows*.

Figura 9. Integração TortoiseSVN/Windows

Para criar esse novo recurso vamos realizar um *branch* e vamos trabalhar em cima dele. Dessa forma o código fonte do *trunk* ainda é o código estável, base para qualquer alteração de emergência. Para criar o *branch* clicamos com o botão direito do mouse sobre a pasta *Projeto* e escolhemos *Tortoise/SVN>Branch,Tag...* Na janela que surgir vamos criar uma nova pasta dentro *branches*, a pasta se chamará *1.1*. Configure as opções como visto na **Figura 10**.

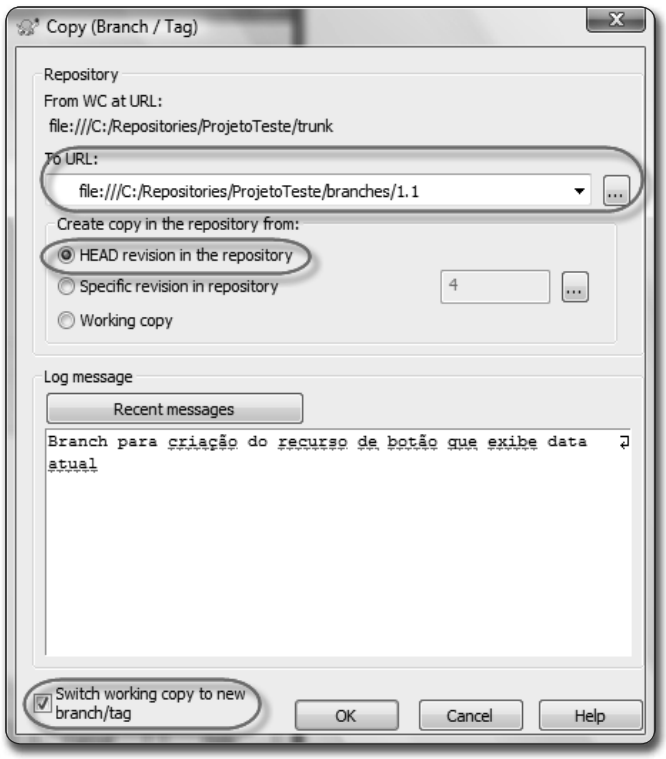


Figura 10. Criando um Branch

Agora é só abrir o projeto no *Delphi* e implementar a funcionalidade requerida pelo usuário do sistema. Para isso vamos mudar o *Caption* do formulário principal para *Versão 1.1*, adicionamos então um *TButton* a ele e no evento *OnClick* do botão lançamos o seguinte código:

```
ShowMessage('Data atua:' + DateToStr(now));
```

Nota

Observe que existe um erro na grafia da mensagem, atual está escrito de forma incorreta. Esse erro é proposital ao exemplo.

A desenvolvedora Carla executou o projeto, achou que está tudo certo e então irá realizar o *check-in* de suas alterações. Para isso acessamos a pasta *Projeto* e veremos que *TortoiseSVN* já destacou quais arquivos estão diferentes do *branch*. Veja **Figura 11**.

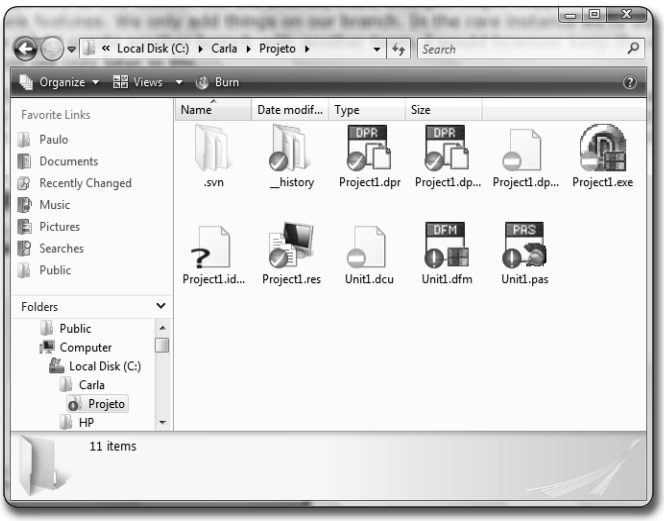


Figura 11. TortoiseSVN exibindo arquivos com diferenças

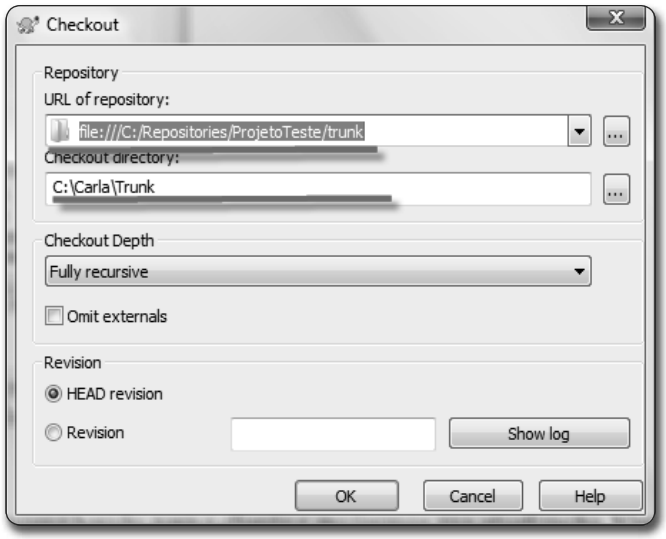


Figura 13. Realizando um check-out a partir do trunk do servidor

Clicamos com o botão direito do *mouse* então sobre a pasta *Projeto* e escolhemos *SVN Commit*. Veja na **Figura 12** o diálogo para realização do *check-in*.

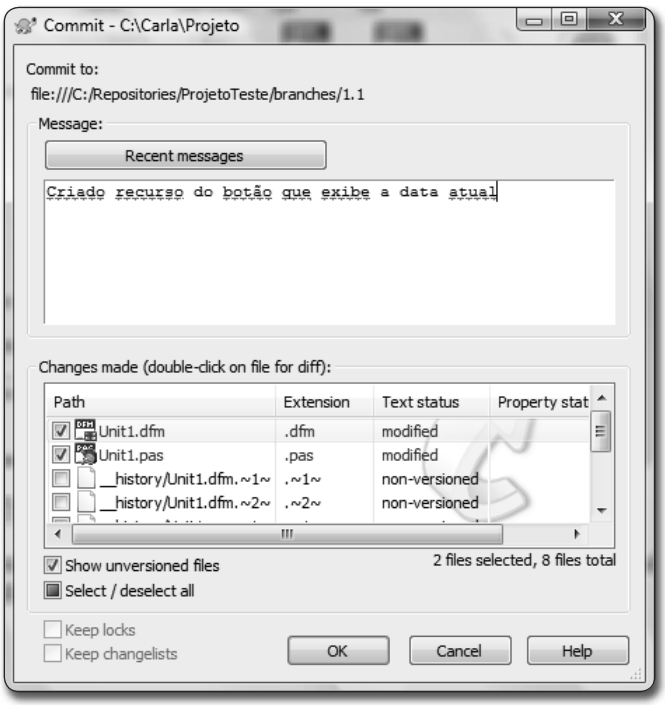


Figura 12. Realizando Check-In

Chegou a hora de fecharmos então uma versão para o cliente e enviarmos sua atualização. Vamos adicionar ao *trunk* a funcionalidade recém criada. Pra isso vamos criar uma subpasta *Trunk* dentro de nossa pasta *Carla* e nela vamos realizar um *check-out* do *trunk* do servidor. Veja **Figura 13**.

E agora vamos realizar o *merge* da nova funcionalidade clicando o botão direito do *mouse* sobre a pasta *Trunk* e escolhendo *TortoiseSVN>Merge*. Na janela de mesclagem escolheremos a opção *Reintegrate a branch*, pois é ela que é responsável em incluir o que fizemos no *branch 1.1*. Então é clicar no *Next* e para finalizar clicar no botão *Merge*. Ao término da mesclagem vemos a confirmação do que foi atualizado.

E agora apenas realizamos um *Commit* de nossa pasta *C:\Carla\Trunk* para o *trunk* do servidor. Podemos até mesmo criar uma *tag* para nossa versão atual. Dessa forma a qualquer momento poderemos restaurá-la. Para isso clicamos com o botão direito do *mouse* sobre a pasta *C:\Carla\Trunk* e escolhemos *TortoiseSVN>Branch/tag...* Na janela *Copy* que surgir especificamos o caminho da *tag*: *file:///C:/Repositories/ProjetoTeste/tags/1.1* no campo *To URL*. Marcamos também a opção *HEAD revision in the repository*, que levará para a nova *tag* todo o código atual do *trunk* e clicamos no *OK* (**Figura 14**).

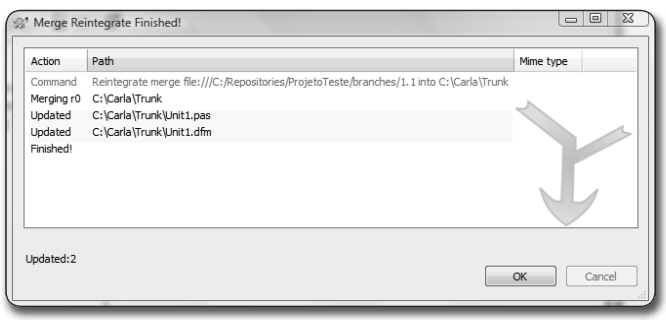


Figura 14. Confirmação do merge

Como terminamos o requisito do cliente, podemos excluir o *branch 1.1* do repositório através da opção *TortoiseSVN>Repo-browser*.

Digamos que o cliente agora solicitou um novo requisito. Incluir mais um botão no formulário principal para exibir desta vez a hora atual. Então o programador Paulo fica responsável por essa funcionalidade. A primeira coisa a se fazer, semelhantemente como que fizemos na primeira funcionalidade, é realizar um *check-out* do *trunk* e criar um novo *branch* para esse recurso.

Com o *branch* criado o Paulo então carrega o projeto no

Delphi e começa a implementar o novo recurso. Mas algo inesperado acontece. Existe um *bug* no *trunk*. O usuário percebeu o erro de grafia da mensagem e solicita urgentemente a correção disso. Então a Carla realiza um novo *Check-out* do *trunk* e cria também um novo *branch* para essa correção.

Observaram uma coisa aqui? A Carla não precisou esperar o Paulo terminar a implementação do recurso do novo botão para poder corrigir o erro, e veja que o erro está na mesna *unit* que ambos vão trabalhar.

Após ter realizado o *branch* e corrigido o problema ela volta a correção para o *trunk*. Mas e quanto ao Paulo? Ele precisa atualizar os fontes dele para pegar essa correção. Senão sem querer ele pode voltar a colocar o *trunk* o mesmo erro. Para evitar isso é preciso criar e utilizar uma regra simples para toda a equipe: *Quem está trabalhando em braches é obrigado que realize um merge do trunk para recuperar possíveis correções*.

Para trazer as correções do *trunk* para o *branch* clicamos com o botão direito do *mouse* sobre a pasta Teste do Paulo e selecionamos *TortoiseSVN>Merge*. Na janela de *merge* escolhemos a opção *Merge two different trees*. Utilizamos essa opção porque precisamos atualizar o código da nossa pasta com o código do *trunk*. Na **Figura 15** vemos como a janela é configurada.

Após ter realizado a mesclagem o programador Paulo já está com a correção para a falha da grafia incluída em sua revisão, portando agora é só realizar o *commit* em seu *branch* e atualizar o *trunk* novamente com as novas funcionalidades.

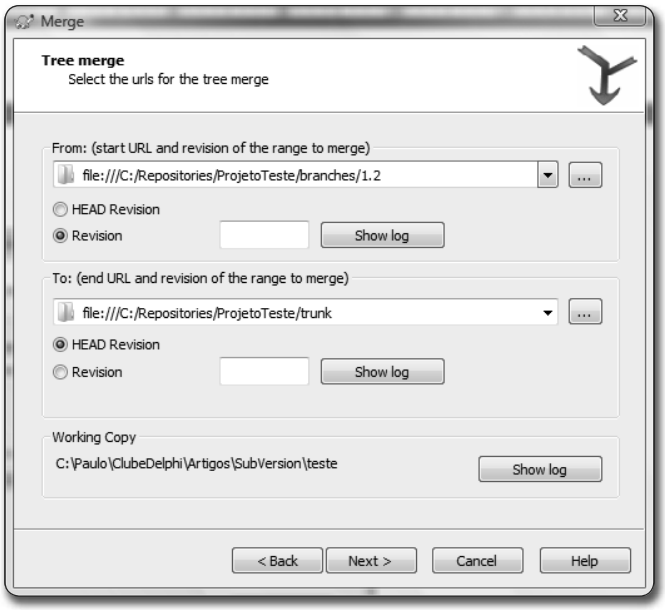


Figura 15. Atualizando o branch

Conclusão

Neste artigo além de conhecer o *Subversion*, pudemos também entender os problemas que a falta de gerência de versões pode trazer para nosso dia a dia e mesmo tendo a ferramenta, conhecendo os problemas é difícil no início implementar esse tipo de controle em um time de desenvolvimento. É difícil porque exige atenção por parte do desenvolvedor de sempre realizar o *branches* ou de realizar o *merge* a partir do *trunk*. Não são poucos os casos onde novos recursos simplesmente desapareceram da versão final do sistema simplesmente porque esqueceram um *merge* e a culpa recai sobre a ferramenta. Portando o mais difícil não é utilizar a ferramenta em si, mas sim, fazer com que os desenvolvedores façam a coisa certa. Abraço. ●

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Cursos Online

Assinatura

ClubeDelphi PLUS

Mais conteúdo DELPHI por muito menos!

A Revista **ClubeDelphi** oferece para seus assinantes uma série de **Cursos Online** de alto padrão de qualidade . Conheça abaixo os cursos já disponíveis..

Curso em destaque

Aplicações Client/Server com dbExpress e Firebird

Confira neste curso online como criar uma aplicação client/server completa no Delphi 7, utilizando dbExpress e Firebird.
Aprenda também: Como trabalhar com um driver dbExpress específico e gratuito para o Firebird ; Como utilizar Orientação a Objetos, e técnicas como herança visual de formulários e relatórios ; saiba como colocar regras de negócios no banco de dados, usando Triggers e Stored Procedures; E crie relatórios com o Quick Report, Rave Reports e Report Builder, e muito mais.

Confira o plano de aula completo:
www.devmedia.com.br/clienteserver

A sua melhor opção de aprendizagem!
Assine a **ClubeDelphi** e comece já seu treinamento!
www.devmedia.com.br/assine

Outros cursos disponíveis: **www.devmedia.com.br/curso**

- Curso Online - Sistema SysCash
- Aplicações client/server com Windows Forms no Delphi 2006
- Criando uma Aplicação multi-camadas Completa com Delphi
- Aplicações WEB com IntraWeb e Delphi 7 (Delphi Win32)

DevMedia GROUP **Mais Informações: www.devmedia.com.br/central - Tel.: 21 3382-5038/ 3382-5025**

Nesta seção você encontra artigos para iniciantes na linguagem Delphi

Crie um editor de textos

Conheça o componente WPTools

Nesse artigo veremos

Como criar um editor de textos semelhante o MS-Word;

Como criar uma rotina de mala direta.

Qual a finalidade

Introduzir o componente WPTools.

Quais situações utilizam esse recurso?

Sistemas que utilizam edição, impressão de contratos, mala direta.

Resumo do DevMan

Editar textos há muito tempo deixou de ser uma característica exclusiva do Microsoft Word e passou a invadir os mais diversos meios. Com a Web 2.0 então, o que temos visto é uma grande variedade de empresas disponibilizando seus editores de texto on-line. Essa necessidade de ter a mão um bom editor passo a fazer parte também da mente dos clientes, que cada vez mais cobram recursos em nossos sistemas.

Desenvolver um editor de textos realmente não é uma tarefa fácil, mas quando se trata de Delphi é possível encontrar os mais variados tipos de componentes no mercado. O WPTools é um exemplo de componente útil e fácil de utilizar. Com ele podemos criar facilmente um processador de textos com diversos recursos.

Vamos construir juntos nesse artigo um pequeno programa para digitação de textos utilizando esses componentes.

Paulo Quicoli

pauloquicoli@devmedia.com.br

Editor técnico da revista ClubeDelphi, formado em processamento de dados pela FATEC-TQ. Atua como analista e desenvolvedor da ControlM Informática (www.controlm.com.br). Utiliza Delphi desde de sua primeira versão, entusiasta do desenvolvimento orientado a objetos, tem publicado vários artigos sobre o assunto. Blog: www.pauloquicoli.spaces.live.com .

A cada dia que passa mais e mais é exigido dos desenvolvedores no que diz respeito às funcionalidades que um sistema pode ter. Um exemplo disso está nos sistemas que precisam, de alguma forma, permitir que um usuário crie, por exemplo, contratos de locação, venda, etc. Isso levará o desenvolvedor a pensar em uma forma de se criar um editor de textos, talvez até mais semelhante ao Word, já que é com ele que o usuário final está mais acostumado.

Então diante da dificuldade, de se criar um editor de texto eficiente o desenvolvedor opta por utilizar o *MS-Word* mesmo, criando uma dependência em seu sistema que não é ideal. É perfeitamente possível utilizar-se de API's específicas para criação e alteração de documentos Word via Delphi, mas essa dependência não é ideal porque prende o sistema ao próprio Word, o que não é uma boa prática. E agora?

Existe uma solução para esse problema,

uma solução totalmente feita em *Delphi*, o componente *WPTools*. Com ele é possível desenvolvermos aplicações bem semelhantes ao Word e ainda ficar desvinculado a eles. Nesse artigo conheceremos os componentes e aprenderemos como utilizá-los.

O que é a WPTools?

É um componente 100% feito em Delphi que permite adicionar ao seu sistemas recursos sofisticados de edição de textos. A *WPTools* trabalha com uma variação do formato de texto *RTF* (*Rich Text Format*) , adaptado para recursos como cabeçalhos e rodapés, estilos para parágrafos, bem como sua justificação e tabelas. O seu download pode ser feito em *www.wptools.com*, inclusive o suporte oferecido é de ótima qualidade. Além de possuir um fórum de discussão muito ativo, existe a possibilidade de se enviar email diretamente para o desenvolvedor do produto e assim receber sua solução em um tempo realmente curto.

Criando um pequeno exemplo

Faça o download do componente e instale em sua versão do Delphi. Após o devido *download* e instalação temos agora uma nova guia na paleta de componentes, como é visto na **Figura 1**.

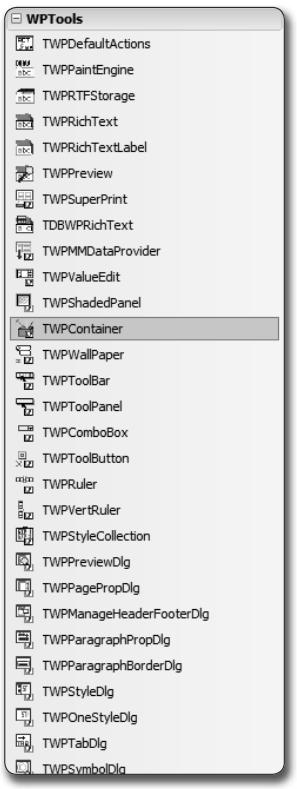


Figura 1. Guia WPTools

Nessa nova guia encontramos tudo o que é necessário para criar um editor de textos similar ao *MS-Word*. Crie uma nova aplicação do tipo *VCL Forms*, salvando-a como desejar. No formulário principal adicione o componente *WPRichText*. Este é o principal controle que compõem o sistema. Clique com o botão direito do *mouse* sobre ele escolha no menu de contexto a opção *QuickConfig*. A janela da **Figura 2** irá aparecer.

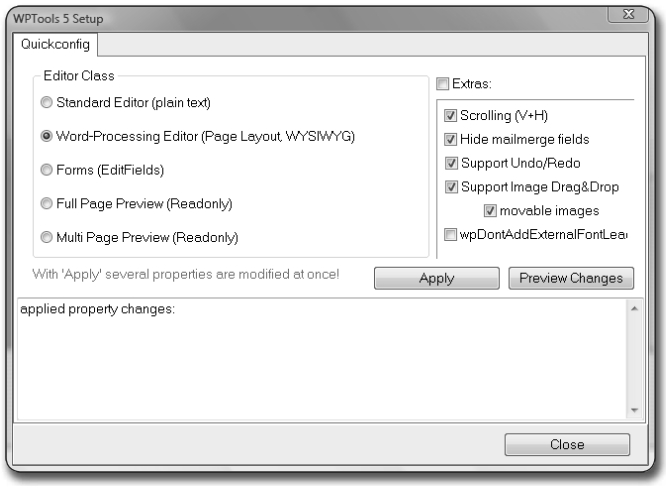


Figura 2. Configuração rápida

Essa janela permite que configuremos de forma rápida as propriedades do controle *WPRichText* para algumas situações específicas como fazê-lo agir como um editor simples, como um *preview* de arquivos apenas ou como um editor completo. Marque a opção *Word-Processing Editor (Page Layout, WYSIWYG)* e clique no botão *Apply*. Com isso feito vamos agora carregar um arquivo *.rtf* qualquer. Se for necessário, crie no *Wordpad* um arquivo.

Adicione um dois botões no formulário, um com o *Caption Carregar arquivo* e o outro *Salvar arquivo*. No evento *onClick* do primeiro botão adicione o seguinte código:

```
  WPRichText1.LoadFromFile('C:\Paulo\
ClubeDelphi\Artigos\WPTools\Documento.rtf', true);
  Form1.Caption := WPRichText1.LastFileName;
```

Primeiramente utilizamos o métodos *LoadFromFile* para carregar um arquivo *.rtf* . Esse método recebe como parâmetros o caminho do arquivo que se deseja carregar e um valor *boolean* que indica se antes de carregar o arquivo o controle *WPRichText* deve ser limpo. Posteriormente utilizamos a propriedade *LastFileName* para exibir no *Caption* do formulário o nome do arquivo que foi aberto. O resultado de tudo isso pode ser visto na **Figura 3**.

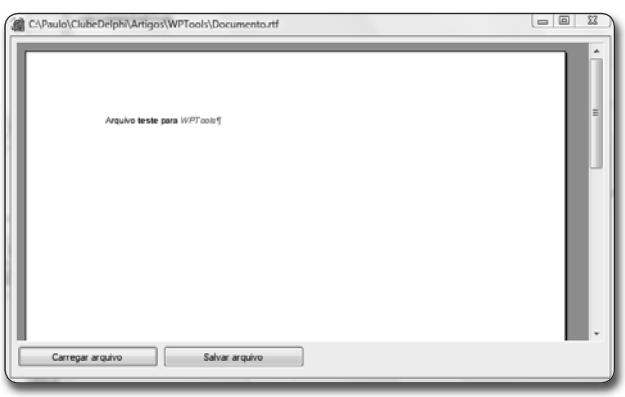


Figura 3. Arquivo pronto para edição

Podemos agora editar o texto e para salvar as alterações vamos incluir no evento *onClick* do segundo botão o seguinte código:

```
WPRichText1.SaveToFile('C:\Paulo\
ClubeDelphi\Artigos\WPTools\Documento.rtf');
```

Veja que estamos salvando as alterações sobre o mesmo arquivo que estamos editando. Mesmo podendo alterar o texto ainda não podemos realizar grandes alterações como mudar o tipo de fonte e cor. Veremos isso a seguir.

Adicionando recursos

Existem diversos recursos que podem ser utlizados com o *WPTools*. Podemos incluir as mais diversas funcionalidades, tais como itálico, negrito, sublinhado, opções de alinhamento,etc. Para incluir tais recursos basta adicionar o componente *TWPToolbar* da guia *WPTools* e ligá-lo ao *WPRichText*. Para ligar os dois controles é preciso apontar a propriedade *UpdateObject* do *TWPToolbar* para o *WPRichText* e fazer o inverso na propriedade *WPToolbar* do *WPRichText*.

Ao executarmos a aplicação veremos agora uma barra de ferramentas que oferece alguns recursos, como pode ser visto na **Figura 4**. Veja que está tudo pronto bastando apenas selecionar o texto e clicar no botão na barra de ferramentas para aplicar a funcionalidade desejada.

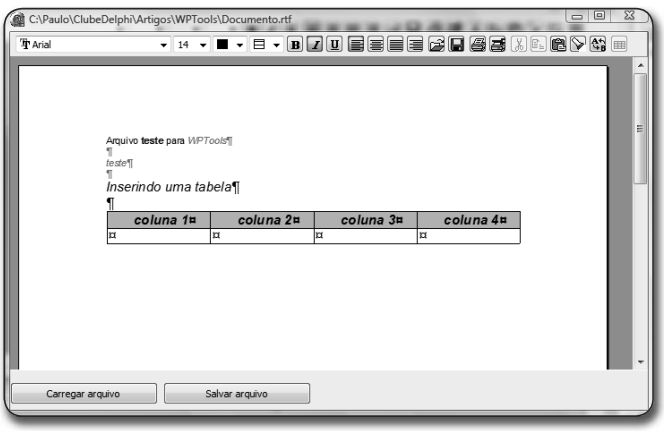


Figura 4. Editor com alguns recursos a mais

Inserindo campos de mesclagem

Do que adianta termos um editor de texto se não podemos gerar, por exemplo, contratos nele onde alguns informações serão obtidas a partir de uma base dados? A *WPTools* permite que criemos nossos campos de mesclagem para posteriormente passarmos os valores, numa espécie de mala direta.

Vamos adicionar um *TListBox* ao fomulário principal e nele adicionar os seguintes itens: *nome*, *endereço*, *bairro*, *cidade* e *uf*. Então no evento *OnDblClick* do *ListBox* vamos fazer com que o item selecionado seja adicionado como um campo de mesclagem. Isso é feito utilizando o método *InputMergeField* do *WPRichText*. O método recebe dois parâmetros. O primeiro é o nome do campo, que ficará registrado na estrutura do arquivo *.rtf*, o segundo é o texto que aparecerá para o usuário. Dessa forma, o evento *OnDblClick* do *ListBox* contém o seguinte código:

```
if ListBox1.ItemIndex >= 0 then
   WPRichText1.InputMergeField(ListBox1.
    Items[ListBox1.ItemIndex],
   ListBox1.Items[ListBox1.ItemIndex]);
```

Execute a aplicação e dê um clique duplo sobre os itens do *ListBox*. O resultado pode ser visto na **Figura 5**.

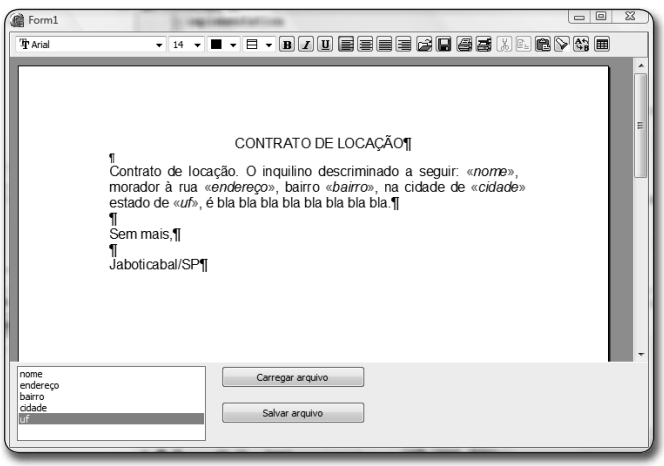


Figura 5. Campos inseridos em um documento

Observe com atenção a **Figura 5**. Veja que além de inserir os campos de mescaglem foi possível também adicionar uma formatação nos mesmos, deixando-os em *itálico*. Um outro detalhe interessante é que automaticamente, ao inserir um campo, a *WPTools* insere uma marcação em sua volta, para facilitar sua identificação. Essa marcação também não é fixa, podemos alterá-la. Adicione no evento *OnCreate* do formulário o seguinte código:

```
WPRichText1.InsertPointAttr.
 CodeOpeningText := '[%N=';
WPRichText1.InsertPointAttr.
 CodeClosingText := ']';
WPRichText1.InsertPointAttr.
 CodeTextColor := clBlue;
```

Execute a aplicação novamente e insira os campos. Você verá que agora a indicação dos campos é feita através de colchetes na cor azul. Veremos agora como passar os valores para campos. Para realizar a mesclagem o componente *WPTools* oferece o método *MergeText* que por sua vez dispara o evento *OnMailMergeGetText*. É nesse evento que colocamos a lógica de como obter os valores, **Listagem 1**.

Listagem 1. Realizando a mesclagem

```
01 procedure TForm1.WPRichText1MailMergeGetText(Sender: TObject;
02   const inspname: string; Contents: TWPMMInsertTextContents);
03 begin
04   if inspname = 'nome' then
05   begin
06     Contents.StringValue := 'Paulo Quicoli';
07     exit;
08   end;
09
10   if inspname = 'endereço' then
11   begin
12     Contents.StringValue := 'Cabo Ulysses Silva Beltrão,
100';
13     exit;
14   end;
15
16   if inspname = 'bairro' then
17   begin
18     Contents.StringValue := 'Centro';
19     exit;
20   end;
21
22   if inspname = 'cidade' then
23   begin
24     Contents.StringValue := 'Jaboticabal';
25     exit;
26   end;
27
28   if inspname = 'uf' then
29   begin
30     Contents.StringValue := 'São Paulo';
31     exit;
32   end;
33 end;
```

Veja que o evento disponibiliza duas variáveis extras, *inspname* que indica qual o campo que está sendo mesclado no momento e *Contents*, que representa o valor que será passado a esse campo. Nós verificamos então qual o campo que será mesclado e passamos o valor apropriado para ele. O caráter de exemplo estamos passando valores fixos, porém em uma aplicação real geralmente vamos obter os valores do campos em algum banco de dados.

Como mencionado antes, para realizar a mesclagem chamamos o método *MergeText*. Vamos adicionar um novo botão ao formulário principal, mudamos seu *Caption* para *Realizar mesclagem* e no seu evento *onClick* colocamos o seguinte código:

```
WPRichText1.MergeText;
```

Na **Figura 6** vemos o texto mesclado.

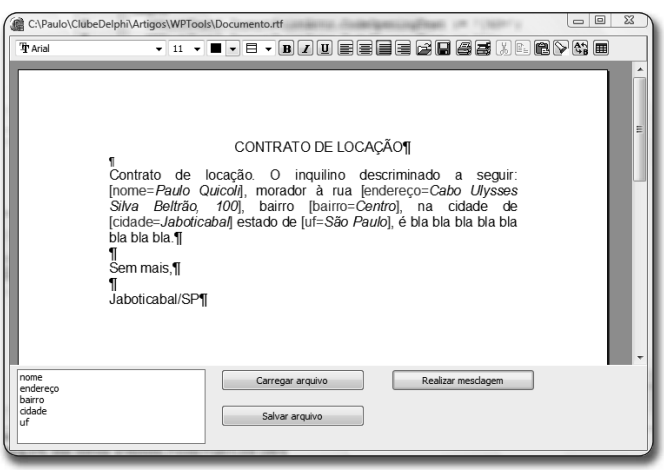


Figura 6. Mesclagem realizada

Se mandarmos imprimir o texto dessa maneira a indicação dos campos também será impressa. Para solucionar esse problema basta *setar* a propriedade *WPRichText1.InsertPointAttr.Hidden* para *true* antes da impressão. Na **Figura 7** vemos o resultado da mesclagem com os indicadores de campos ocultos.

Como podemos ver, há diversos recursos que podem ser utilizados de maneira muito simples.

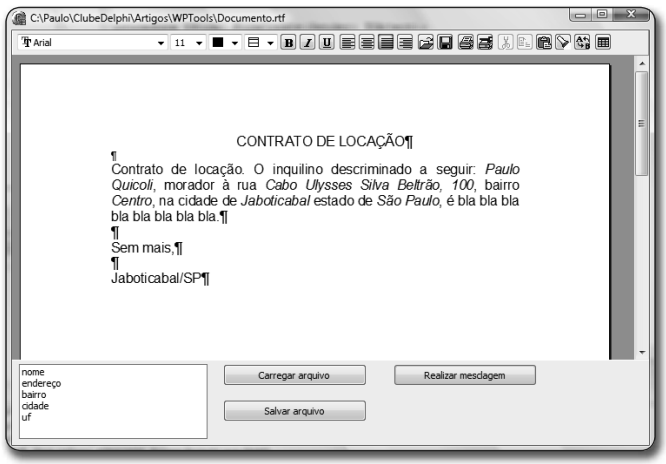


Figura 7. Resultado da mesclagem com os indicadores ocultos

Até agora criamos nosso próprio editor de textos com o mínimo de código possível, mas ainda deixamos de lado recursos como configuração de margens, papel, cabeçalhos e rodapé. Tudo isso fica fácil de implementar ao utilizar o editor completo disponibilizado pela WPTools.

Utilizando o editor padrão

A forma mais rápida para se utilizar o editor padrão é copiar os seguintes arquivos, que fazem parte do código fonte da *WPTools*: *WPDefActions.dfm*, *WPDefActions.pas*, *WPDefEditor.dfm* e *WPDefEditor.pas* para a pasta do nosso exemplo e adicioná-los ao projeto. Depois adicionamos um botão no formulário principal e mudamos seu *Caption* para *Chamar editor padrão*. No seu evento *onClick* apenas instanciamos o editor padrão, lembrando de adicionar uma referência no *uses* à *WPDefEditor*.

```
WPToolsEditor := TWPToolsEditor.Create(nil);
WPToolsEditor.ShowModal;
FreeAndnIl(WPToolsEditor);
```

Ao executarmos o aplicativo vamos ver que agora temos um editor completo, com os mais diversos recursos, como visto na **Figura 8**.

E como copiamos para nosso projeto os fontes do editor, podemos alterá-los de acordo com nossa necessidade. E uma necessidade é traduzir o editor para o português. Como pode ser visto na **Figura 8** os textos e unidades de medidas dos diálogos estão em inglês.

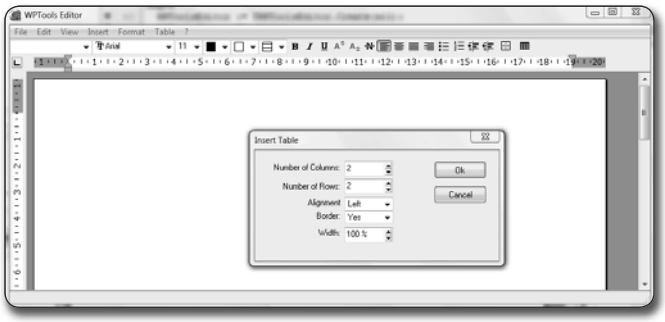


Figura 8. Editor padrão da WPTools

Traduzindo a WPTools

Vamos primeiramente ajustar a localização do componente. Quando digo localização quero dizer sobre a unidade de medidas utilizadas. Vamos ajustar para que seja utilizado centímetros ao invés de milímetros. No evento *onCreate* do editor padrão (*wpDefEditor*) incluímos o código:

```
GlobalValueUnit := euCm;
```

• *euCm:* é uma constante que representa a unidade *centímetro* e ao atribuí-la à variável global *GlobalValueUnit* a atualização é imediata. Veja a **Figura 9**.

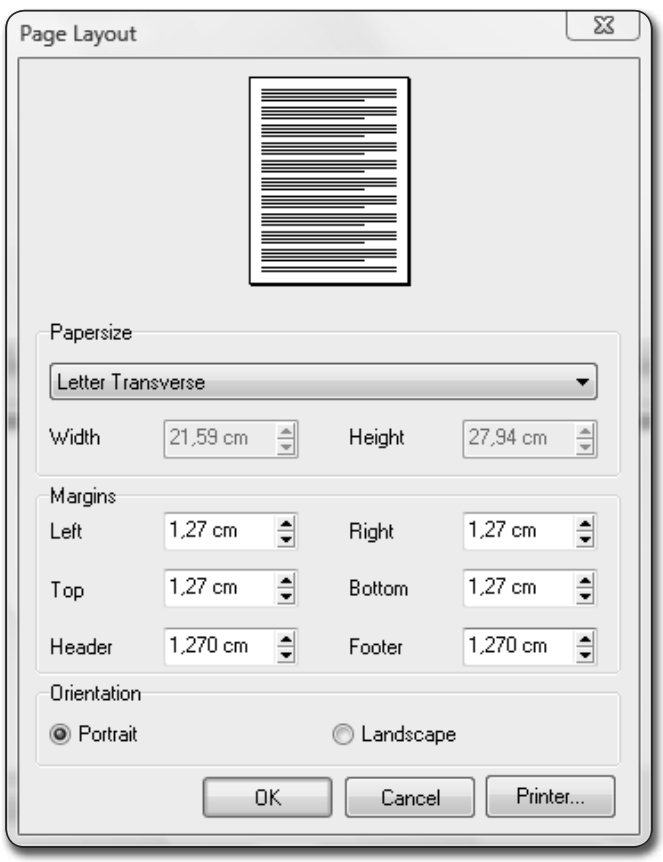


Figura 9. Centímetros sendo utilizado

Após isso vamos traduzir os diálogos. Para isso existe o componente *WPLanguageControl* que pode carregar um arquivo *XML* contendo a tradução desejada. Esse componente não faz parte do pacote padrão de instalação. Para adicioná-lo, para adicionar ao pacote a *unit WPShared.pas* e *recompilar*. Após termos adicionado o novo componente ao pacote da *WPTools* vamos então colocá-lo no formulário padrão do projeto e adicionar o arquivo *XML* com a tradução para o português do Brasil. Damos um clique duplo sobre o componente e na janela que surgir acessamos o menu *File>Load*. O arquivo de tradução está distribuído dentro da pasta que contém os demos da *WPTools*, exatamente em *<Diretório de instalação>\Demos\Tasks\Localisation*. Na **Figura 10** vemos o componente com a tradução carregada.

Nota do DevMan

XML (eXtensible Markup Language) e é nada mais que uma linguagem de marcação largamente utilizada na Web. Seu formato simples acabou agradando desenvolvedores de diferentes linguagens, pois possui o conceito de Tags para organizar as informações em seu interior. Dessa forma fica bastante simples armazenar boa quantidade de informação em arquivos desse formato.

O XML possui outras variantes como o XLST que é basicamente um formato de folha de estilos para o XML. Em Delphi podemos utilizar o XML como base de dados já que seu formato possibilita a criação de campos (fileds) e registros que podem mais tarde serem utilizados como tabelas.

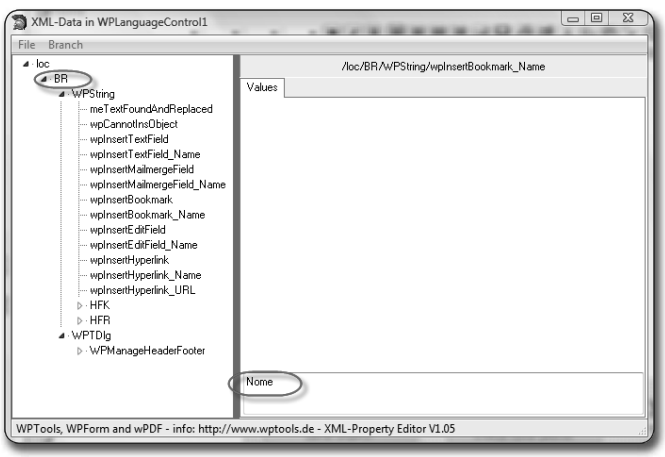


Figura 10. Tradução carregada

Para aplicar a tradução, no evento *onCreate* do formulário principal adicionamos o código:

```
WPLangInterface := TWPLocalizationInterface.
 Create(WPLanguageControl1);
WPLanguageControl1.GlobalLanguage := 'BR';
WPLocalizeLoadForms := TRUE;
WPTools_LoadVCLStrings;
WPTools_LoadActionStrings;
```

E no evento *onDestroy*:

```
WPLangInterface.Free;
```

É preciso também adicionar na seção *uses* as *units WPUtil e WPActnStr*. Execute o editor padrão agora e veja o texto em português, **Figura 11**.

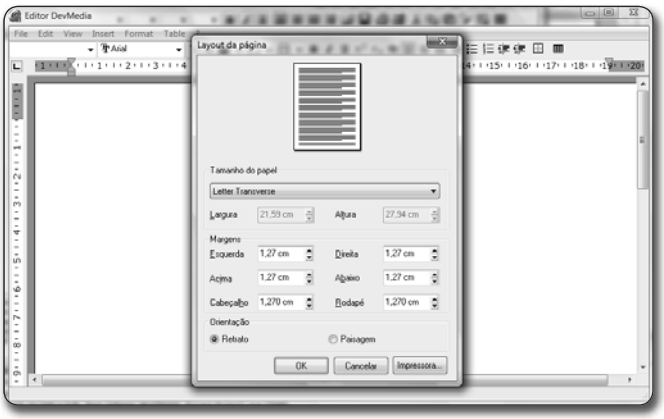


Figura 11. Diálogo de configuração da página em português

Nem todo o texto foi traduzido, como pode ser visto. Mas lembre-se que copiamos os fontes do editor padrão para nosso projeto, portanto, podemos nós mesmos traduzir os menus.

Como vimos, os componentes *WPTools* são bastante flexíveis e acrescentam muitos recursos as nossas aplicações.

Conclusão

Uma das grandes vantagens do *Delphi* no desenvolvimento 32bits é sua enorme gama de componentes que são disponibilizados por empresas especialistas no assunto. Nem todos são gratuitos, mas mesmo assim com o *Delphi* nunca ficamos na mão. Prova disso é a *WPTools*, que nos permite construir quase um *MS-Word* com o mínimo de código possível. Abraço! ●

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Cursos Online

Assinatura
java magazine PLUS
Mais conteúdo JAVA por muito menos!

A Revista **Java Magazine** oferece para seus assinantes uma série de Cursos Online de alto padrão de qualidade .

Conheça abaixo o curso já disponível:
www.devmedia.com.br/curso/javamagazine

Introdução ao desenvolvimento para celulares com J2ME

A sua melhor opção de aprendizagem!

Assine a **Java Magazine** e
Comece já seu treinamento!
www.devmedia.com.br/assine

DevMedia GROUP Mais Informações: www.devmedia.com.br/central - Tel.: 21 2220-5375 / 2220-5435

Nesta seção você encontra artigos sobre a linguagem PHP e a ferramenta Delphi for PHP

Gerando miniaturas em PHP

Utilize thumbnails em seus Web Sites PHP

Nesse artigo veremos
Uso da biblioteca GD;
Criação de thumbnails, miniaturas;

Qual a finalidade
A principal finalidade é diminuir o tempo de carga de uma página de fotos.

Quais situações utilizam esse recurso?
Todo e qualquer web site que se utilize de muitas fotos pode ter esse recurso aplicado e, com isso, ganhar em performance e velocidade.

Adriano Santos
falecom@adrianosantos.pro.br
É desenvolvedor Delphi desde 1998, professor e programador PHP, bacharel em Comunicação Social pela Universidade Cruzeiro do Sul, SP. É Editor Chefe das revistas ClubeDelphi e WebMobile. Gerente de TI na SoftPark, parceira Borland, membro fundador do DUG-SP (www.dug-sp.com) e mantenedor do blog Delphi to Delphi (www.delphitodelphi.blogspot.com) com dicas, informações e tudo sobre desenvolvimento Delphi.

Mesmo com o crescimento da banda larga em todo o mundo, economizar alguns bytes e largura de banda é a meta de muitos provedores, servidores e desenvolvedores de site. Isso porque quanto mais rápido um site carrega, melhor é a percepção do internauta em relação ao *website* a que se visita. É fato que ninguém gosta de ficar minutos na frente de um computador esperando que todas as imagens da *homepage* sejam carregadas.

Hoje em dia, aliás, já há muito tempo usa-se *Thumbnails* para melhorar a velocidade de carregamento de uma página. Os *Thumbnails* são miniaturas das imagens originais como podemos visualizar na **Figura 1** retirada de um portal de notícias. O próprio *Windows* utiliza esse tipo de recurso quando selecionamos a opção *Exibir Miniaturas* no *Windows XP* ao clicar com o botão direito em uma pasta ou *Ícones Médios, Grandes* e *Extra Grandes* no *Windows Vista* (**Figura 2**).

Resumo do DevMan

O esforço em aumentar a banda larga em todo o país tem sido bastante grande por parte das empresas que provêem o serviço. Porém, enquanto não conseguimos velocidades altíssimas, é importante que nos preocupemos com o carregamento das páginas, a velocidade e qualidade de navegação.

Nesse artigo veremos como criar as famosas Thumbnails, miniaturas de figuras/fotos em álbuns de fotografias.

Figura 1. Exemplo de Thumbnails em portal de notícias

O *Windows* utiliza esse recurso para aumentar a velocidade de carregamento das imagens de determinada pasta. Imagine na *Web* você abrir um site que possui *10* ou *20* imagens de *1mb* cada uma. Certamente o tempo de carga será muito alto e o internauta talvez desista de continuar acessando

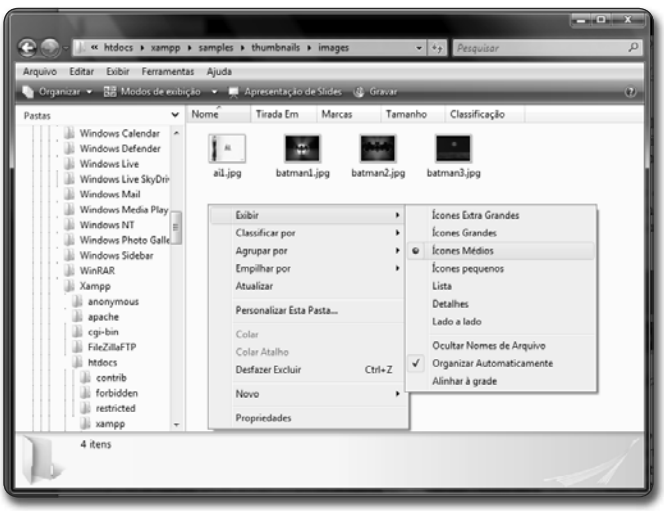


Figura 2. Exibindo miniaturas no Windows Vista

Existem algumas formas de se criar *Thumbnails*. Uma delas é criar cópias das imagens originais e reduzi-las utilizando algum programa de edição de imagens como o *Adobe Photoshop*, por exemplo. Porém, se estamos criando um álbum de fotografias, por exemplo, e queremos ter a possibilidade de enviar fotos a todo instante, alterar o tamanho de cada foto a todo instante torna-se inviável.

Nesse artigo, veremos como criar *Thumbnails* utilizando a biblioteca *GD* do *PHP* que por sinal facilita bastante o nosso trabalho. Mão na massa.

Considerações iniciais

Como dito anteriormente, utilizaremos a biblioteca *GD* do *PHP*. A biblioteca nada mais é do que uma *DLL* que possui as funções necessárias para a criação da miniatura. Para habilitar a *DLL* em seu *PHP*, é necessário abrir o arquivo *PHP.INI*, normalmente localizado no diretório *System32* do *Windows* e logo após localizar a seguinte linha:

```
#extension=php_gd2.dll
```

Perceba que temos um sinal # na frente da linha, indicando que a mesma encontra-se comentada. Basta então retirar o sinal e salvar o arquivo *.ini*. Feito isso já estamos habilitados para usar *GD* em nossas aplicações *PHP*.

Na edição *101* de *ClubeDelphi*, publiquei um artigo falando como configurar um bom ambiente de testes para *PHP* utilizando o *Xampp*. Caso esteja usando esse ambiente, não é necessário fazer a configuração, pois por padrão o *GD* já encontra-se ativado, como vemos na **Figura 3**.

gd

GD Support	enabled
GD Version	bundled (2.0.34 compatible)
FreeType Support	enabled
FreeType Linkage	with freetype
FreeType Version	2.1.9
T1Lib Support	enabled
GIF Read Support	enabled
GIF Create Support	enabled
JPG Support	enabled
PNG Support	enabled
WBMP Support	enabled
XBM Support	enabled

Figura 3. Relatório phpinfo() mostrando GD ativado

Criando o arquivo de script

Vamos iniciar criando e entendendo o arquivo de *script* responsável por criar as miniaturas. Para esse exemplo precisaremos apenas de dois arquivos. Um contendo o *script* das miniaturas e o outro para exibição das imagens. Abra então o bloco de notas do Windows ou qualquer outro editor de textos para que possamos digitar o arquivo PHP.

A primeira coisa que faremos no script é definir as constantes de *Endereço das Imagens*, *Largura Máxima* e *Altura Máxima*. Para isso usaremos o método *define()*. Comece o *script* digitando as linhas de programação abaixo:

```
<?PHP
define('PATH_IMG', '<caminho>');
define('MAX_WIDTH', 160);
define('MAX_HEIGHT', 120);
...
```

Veja que na definição da constante *PATH_IMG* coloquei propositalmente o texto *<caminho>*. Pois nesse momento podemos atribuir o local correto das imagens de nosso *WebSite*. Precisamos apenas trocar *<caminho>*, pelo caminho certo das fotos. Veja um exemplo:

```
define('PATH_IMG', 'C:\Xampp\exemplos\
thumbnails\images');
```

Aqui estamos apontando para a pasta *images* presente no diretório da aplicação. É importante colocar as fotos em

Figura 4. Topo do arquivo index.php

locais diferentes, pois faremos uma varredura do diretório transformaremos em *thumbnails* todos os arquivos *jpg* e *jpeg* encontrados. Isso significa que se tivermos outras imagens no mesmo diretório, como os ícones que compõem a aplicação em si, elas também serão convertidas.

A próxima etapa do *script* é responsável por receber o nome do arquivo de imagem, retirar o símbolo *"..."*, caso haja, e concatenar com a constante *PATH_IMG*. Por fim atribuímos esse endereço de imagem a variável *$image_path*. A última linha desse trecho atribui o valor *null* a variável *$img*. Não chega a ser a criação e/ou declaração da variável, assim como fazemos em Delphi, mas é uma forma de iniciá-la. Digite o código a seguir no mesmo arquivo que estamos criando.

```
# Pega onde está a imagem
$image_file = str_replace('..', '', $_
SERVER['QUERY_STRING']);
$image_path = PATH_IMG . '/' . $image_file;
# Carrega a imagem
$img = null;
```

Perceba o uso da função *str_replace* no início. Essa função faz a substituição de um dado caractere por outro. Nesse caso estamos informando à função que o *".."* encontrado na *QUERY_STRING*, deve ser substituído por *''*, ou seja, nenhum caractere. Em Delphi temos essa função também sob o nome de *ReplaceString*. Em outras palavras, nosso *script* receberá algo como a linha a seguir:

```
<img src="thumb.php?pasta/foto.jpg">
```

Nossa constante receberá o caminho físico das fotos e o concatenará com o nome da foto recebido na *QueryString*. Dessa forma teremos algo como vemos a seguir. Com isso em "mãos" o *script* é capaz de transformar em *thumbnails*.

```
$image_path = 'C:\Xampp\exemplos\
thumbnails\images\image.jpg';
```

Agora precisamos tomar outra providência. Nem todos os tipos de imagem podem ser convertidos em *miniaturas* e além do mais, cada tipo de extensão precisa de um tratamento especial. Por isso, extraímos a extensão do arquivo e o tratamos chamando a função adequada. Veja na **Listagem 1**.

Listagem 1. Analisando a extensão e tipo de arquivo

```
$extensao = strtolower(end(explode('.',$image_path)));

if ($extensao == 'jpg' || $extensao == 'jpeg') {
  $img = @imagecreatefromjpeg($image_path);
}elseif ($extensao == 'png') {
  $img = @imagecreatefrompng($image_path);
}elseif ($extensao == 'gif') {
$img = @imagecreatefromgif($image_path);
}
```

A variável *$extensao* está recebendo o resultado de várias funções, uma validando a outra. Primeiro separamos literalmente o nome do arquivo e sua extensão usando o método *explode*, que na verdade o transforma em um *array*. Enviamos como parâmetro para ele um ponto, que é o separador entre nome e extensão como já estamos acostumados. Depois pegamos o último elemento do *array*, nesse caso a extensão, usando o método *end*. Por fim, transformamos tudo em minúsculo com a função *strtolower*.

Na seqüência testamos cada valor e chamamos as respectivas funções para criação das miniaturas. Veja, se a imagem for um *jpg* ou *jpeg*, a função *@imagecreatefromjpeg* é chamada. Case seja *png*, então chamamos *@imagecreatefrompng* e assim sucessivamente. Evidentemente que não há todas aqui, mas basta uma boa lida na documentação do PHP para encontrar as demais funções. Repare também que o resultado da função é passado diretamente para variável *$img*.

Vamos continuar a codificação do *script*. Após criar a imagem, precisamos testar se ela foi criada com sucesso e então começar a manipular suas propriedades. Veja o código da **Listagem 2**. Logo de início pegamos o valor de largura (*$width*), altura (*$height*) e sua escala (*$scale*). Com isso analisamos a imagem. Se a escala for maior que o permitido, então encolhemos a imagem (*Linha 5* a *7*). Criamos então uma imagem temporária na linha *8* e copiamos para ela (*Linha 9*) o mesmo tamanho da imagem original. Destruímos a imagem que estava gravada em *$img* e atribuímos a ela a imagem temporária.

Listagem 2. Tratamento da imagem

```
...
01 if ($img){
02   $width = imagesx($img);
03   $height = imagesy($img);
04   $scale = min(MAX_WIDTH / $width, MAX_HEIGHT / $height);
05   if ($scale < 1) {
06     $new_width = floor($scale * $width);
07     $new_height = floor($scale * $height);
08     $tmp_img = imagecreatetruecolor($new_width, $new_height);
09     imagecopyresized($tmp_img, $img, 0, 0, 0, 0, $new_width,
10       $new_height, $width, $height);
11     imagedestroy($img);
12     $img = $tmp_img;
13   }
14 }
```

Caso ocorra algum erro, então criamos uma imagem de erro para mostrar ao internauta que algo deu errado no site. Veja a **Listagem 3**. Os métodos *imagecolorallocate* e *imageline* são responsáveis por gerar uma pequena imagem no lugar da miniatura, demonstrando que algo aconteceu.

Listagem 3. Criando uma imagem de erro

```
..
if (!$img){
  $img = imagecreate(MAX_WIDTH, MAX_HEIGHT);
  imagecolorallocate($img, 204, 204, 204);
  $c = imagecolorallocate($img, 153, 153, 153);
  $c1 = imagecolorallocate($img, 0, 0, 0);
  imageline($img, 0, 0, MAX_WIDTH, MAX_HEIGHT, $c);
  imageline($img, MAX_WIDTH, 0, 0, MAX_HEIGHT, $c);
  imagestring($img, 2, 12, 55, 'erro ao carregar imagem', $c1);
}
..
Para fechar o script, mostramos a imagem usando o método
imagejpeg. O método header descreve qual será o tipo de conteúdo
da página atual.

..
header('Content-type: image/jpeg');
imagejpeg($img);
?>
```

Salve esse arquivo de script com o nome de *thumbs.php*. O código completo do arquivo pode ser visto na **Listagem 4**.

Listagem 4. Código de thumbs.php

```
<?PHP
define('PATH_IMG', '<caminho>');
define('MAX_WIDTH', 160);
define('MAX_HEIGHT', 120);
# Pega onde está a imagem
$image_file = str_replace('..', '', $_SERVER['QUERY_STRING']);
$image_path = PATH_IMG . '/' . $image_file;
# Carrega a imagem
$img = null;

$image_path = 'C:\Xampp\exemplos\thumbnails\images\image.jpg';

$extensao = strtolower(end(explode('.',$image_path)));

if ($extensao == 'jpg' || $extensao == 'jpeg') {
  $img = @imagecreatefromjpeg($image_path);
}elseif ($extensao == 'png') {
  $img = @imagecreatefrompng($image_path);
}elseif ($extensao == 'gif') {
$img = @imagecreatefromgif($image_path);
}

if ($img){
  $width = imagesx($img);
  $height = imagesy($img);
  $scale = min(MAX_WIDTH / $width, MAX_HEIGHT / $height);
  if ($scale < 1) {
    $new_width = floor($scale * $width);
    $new_height = floor($scale * $height);
    $tmp_img = imagecreatetruecolor($new_width, $new_height);
    imagecopyresized($tmp_img, $img, 0, 0, 0, 0, $new_width,
      $new_height, $width, $height);
    imagedestroy($img);
    $img = $tmp_img;
  }
}

if (!$img){
  $img = imagecreate(MAX_WIDTH, MAX_HEIGHT);
  imagecolorallocate($img, 204, 204, 204);
  $c = imagecolorallocate($img, 153, 153, 153);
  $c1 = imagecolorallocate($img, 0, 0, 0);
  imageline($img, 0, 0, MAX_WIDTH, MAX_HEIGHT, $c);
  imageline($img, MAX_WIDTH, 0, 0, MAX_HEIGHT, $c);
  imagestring($img, 2, 12, 55, 'erro ao carregar imagem', $c1);
}

header('Content-type: image/jpeg');
imagejpeg($img);
?>
```

Mostrando as miniaturas

Sem dúvida essa é a parte mais simples. Aqui basta que saibamos como varrer o diretório e chamar o arquivo *thumbs.php* para então mostrar as miniaturas no *browser*. Não seria necessário, mas para deixarmos a tela um pouco mais interessante, criei um cabeçalho no *Paint* para inserir em nosso *Web Site* (**Figura 4**).

Crie um novo arquivo e salve-o com o nome de *index.php* no mesmo diretório onde salvou *thumbs.php*. O código completo da página pode ser visto na **Listagem 5**. Se estiver usando o bloco de notas para edição do exemplo, digite apenas o código a seguir abaixo da *tag <body>* e salve o arquivo de cabeçalho no mesmo diretório onde salvou os dois outros arquivos.

```
<img src="top.jpg" />
<br>
```

Isso faz com que o cabeçalho seja exibido, e uma linha de espaço envie o conteúdo para baixo. Se estiver usando um editor de *HTML*, como o *Dreamweaver*, por exemplo, localize alguma função do tipo *Inserir>Imagem*. Das linhas *1* até *19*, estamos definindo as características do nosso arquivo *HTML* e que não serão comentadas por não fazer parte do escopo desse artigo. O que nos interessa está entre as linhas 23 e 35. Nelas estamos montando um laço *while* que varre o diretório de imagens e, a cada interação, envia a foto selecionada para o arquivos *thumbs.php* que por sua vez transforma-a em miniatura. Veja as linhas 30, 31 e 32. Elas dão um *print*, ou seja, imprimem a fotografia na tela. Veja o resultado final na **Figura 5**.

Listagem 5. Código da página index.php

```
01 <!doctype html public "-//W3C//DTD XHTML 1.0 Transitional//EN"
02 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
04 <head>
05 <title>CD103 - Criando Thumbnails em PHP</title>
06 <style type="text/css" media="all">
07 <!--
08   body, p {
09     font-family: arial, sans-serif;
09   font-size: 12px;
10   }
11   image   {
12     border: 0;
13    display: block;
14    clear: both;
15    margin: 15px;
16    text-align: center;
16   }
17 -->
18 </style>
19 </head>
20 <body>
21 <img src="top.jpg" />
22 <br />
23 <?php
24   //Abre o diretório atual para leitura de imagens...
25   if ($handle = opendir('<caminho>\images')) {
26     while (false !== ($file = readdir($handle))) {
27       if (strtolower(substr($file, -4, 4)) == '.jpg' ||
28         strtolower(substr($file, -5, 5)) == '.jpeg'){
29       $alt = 'clique para ver a imagem no tamanho original...';
30       print '<a href="'.'./images/'.$file.'" target="_blank">
31         <img src="thumb.php?'.$file.
32         '" border=0"' . '" alt="'.$alt.'" /></a>'."\n";
33     }
34   }
35   closedir($handle);
36 }
37 ?>
38 <p> </p>
39 </body>
40 </html>
```

Figura 5. Exemplo funcionando com os thumbnails

Como podemos ver, não há grandes segredos no desenvolvimento de aplicações que se utilizem de *Thumbnails*, miniaturas. O mais importante é que agora temos uma página carregando em grande velocidade.

Conclusão

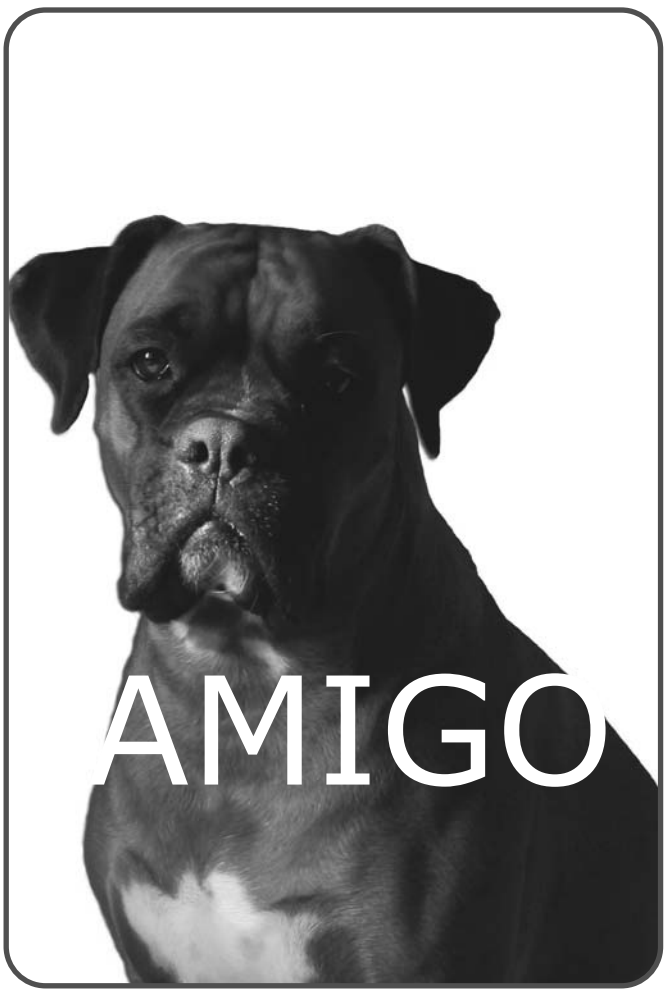
O desenvolvimento de web sites no geral, tem evoluído significativamente. As tecnologias e linguagens de programação têm melhorado cada vez mais, e certamente com isso todos nós ganhamos. Com apenas o uso da biblioteca GD criamos um sistema capaz de exibir *thumbnails*, pequenas imagens que representam as fotos originais de um álbum de fotografias. Há outras bibliotecas capazes de realizar as mais diversas tarefas do dia-a-dia. Confira a documentação do PHP. Até a próxima e boa sorte. ●

Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback



Existem coisas que não conseguimos ficar sem!

...só pra lembrar, sua assinatura pode estar acabando!

Renove Já!

www.devmedia.com.br/renovacao

Para mais informações: www.devmedia.com.br/central

DevMedia GROUP

A infra-estrutura de TI que você precisa.

A Alog trabalha 24 horas por dia, 7 dias por semana, 365 dias por ano para garantir aos seus clientes corporativos estabilidade, agilidade, redundância, escalabilidade, e mais do que tudo, **tranqüilidade**. Entre em contato e saiba por que a Alog é líder em hosting gerenciado

ALOG
DATA CENTERS DO BRASIL

HOSTING GERENCIADO | COLOCATION | CONTIGÊNCIA
E-MAIL CORPORATIVO | CONECTIVIDADE | SERVIÇOS PROFISSIONAIS

0800 282 3330 | alog.com.br

A INVASÃO DE PROPRIEDADE NÃO SE LIMITA A RESIDÊNCIAS.

Assegure-se que seu software está protegido pelo líder em segurança da informação, antes que seja tarde demais.

SafeNet tem à sua disposição a proteção de software mais avançada e segura. As Chaves de Hardware Sentinel oferecem tecnologia que incorpora criptografia de chave pública, criptografia AES e autenticação interna para garantir o mais alto nível de proteção anti-pirataria.

Peça seu Kit de Desenvolvimento hoje mesmo.

Entre em contato em +55 11 4208-7700 em SP-Brasil
ou por e-mail infopirataria@safenet-inc.com

25 YEARS
1983–2008

SafeNet